Exhibit A

to

Complaint for Patent Infringement

The ’442 Patent

US008327442B2

(12) **United States Patent**
Herz et al.

(10) **Patent No.: US 8,327,442 B2**
(45) **Date of Patent: Dec. 4, 2012**

(54) # SYSTEM AND METHOD FOR A DISTRIBUTED APPLICATION AND NETWORK SECURITY SYSTEM (SDI-SCAM)

(76) Inventors: **Frederick S. M. Herz**, Warrington, PA (US); **Walter Paul Labys**, Salt Lake City, UT (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 315 days.

(21) Appl. No.: **10/746,825**

(22) Filed: **Dec. 24, 2003**

(65) **Prior Publication Data**

US 2006/0053490 A1 Mar. 9, 2006

Related U.S. Application Data

(63) Continuation-in-part of application No. 10/693,148, filed on Oct. 23, 2003, now abandoned.

(60) Provisional application No. 60/436,363, filed on Dec. 24, 2002.

(51) **Int. Cl.**
G06F 11/30 (2006.01)

(52) **U.S. Cl.** **726/23**; 726/22; 726/24; 726/25; 713/188

(58) **Field of Classification Search** 726/22–25; 713/182, 188
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,088,804	A *	7/2000	Hill et al.	726/25
6,122,743	A *	9/2000	Shaffer et al.	726/3
6,158,010	A *	12/2000	Moriconi et al.	726/1
6,301,668	B1 *	10/2001	Gleichauf et al.	726/25
6,345,299	B2 *	2/2002	Segal	709/229
6,347,338	B1 *	2/2002	Segal	709/229
6,353,385	B1 *	3/2002	Molini et al.	340/506
6,405,318	B1 *	6/2002	Rowland	726/22
6,530,024	B1 *	3/2003	Proctor	726/23
6,597,777	B1 *	7/2003	Ho	379/133
6,663,000	B1 *	12/2003	Muttik et al.	235/375
6,711,615	B2 *	3/2004	Porras et al.	709/224
6,725,377	B1 *	4/2004	Kouznetsov	726/23
6,769,066	B1 *	7/2004	Botros et al.	726/23
6,772,349	B1 *	8/2004	Martin et al.	726/22
6,952,779	B1 *	10/2005	Cohen et al.	726/22
6,996,843	B1 *	2/2006	Moran	726/23
7,013,330	B1 *	3/2006	Tarbotton et al.	709/219
7,058,968	B2 *	6/2006	Rowland et al.	726/1
7,181,769	B1 *	2/2007	Keanini et al.	726/23
7,185,368	B2 *	2/2007	Copeland, III	726/25
7,222,366	B2 *	5/2007	Bruton et al.	726/23
7,228,564	B2 *	6/2007	Raikar et al.	726/23
7,594,260	B2 *	9/2009	Porras et al.	726/13

(Continued)

OTHER PUBLICATIONS

Denning, Dorothy E., "An Intrusion-Detection Model", IEEE Transactions on Software Engineering, vol. Se-13, No. 2, Feb. 1987, pp. 222-232.*

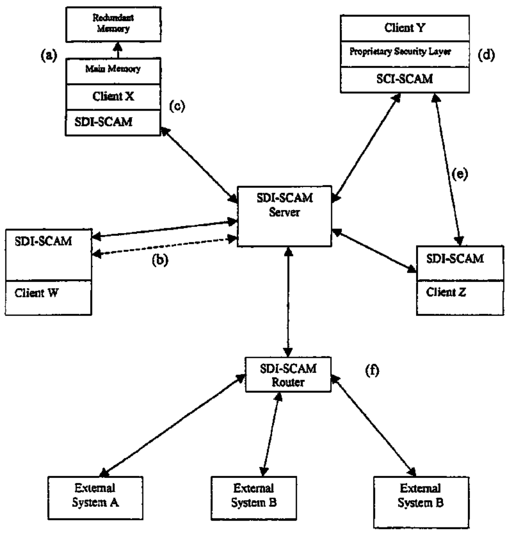
(Continued)

Primary Examiner — Yin-Chen Shaw
(74) *Attorney, Agent, or Firm* — Woodcock Washburn LLP

(57) **ABSTRACT**

This document discloses the architecture and proposed application of a highly distributed network security system. Using a combination of intelligent client-side and server-side agents, redundant memory arrays, duplicate network connections, and a variety of statistical analytics, which are cleverly designed to anticipate, counteract and defeat likely strategic designs, behaviors and adaptations of these threats which may be intended to evade or even disable the network security system, this system serves to detect, prevent, and repair a wide variety of network intrusions.

33 Claims, 1 Drawing Sheet



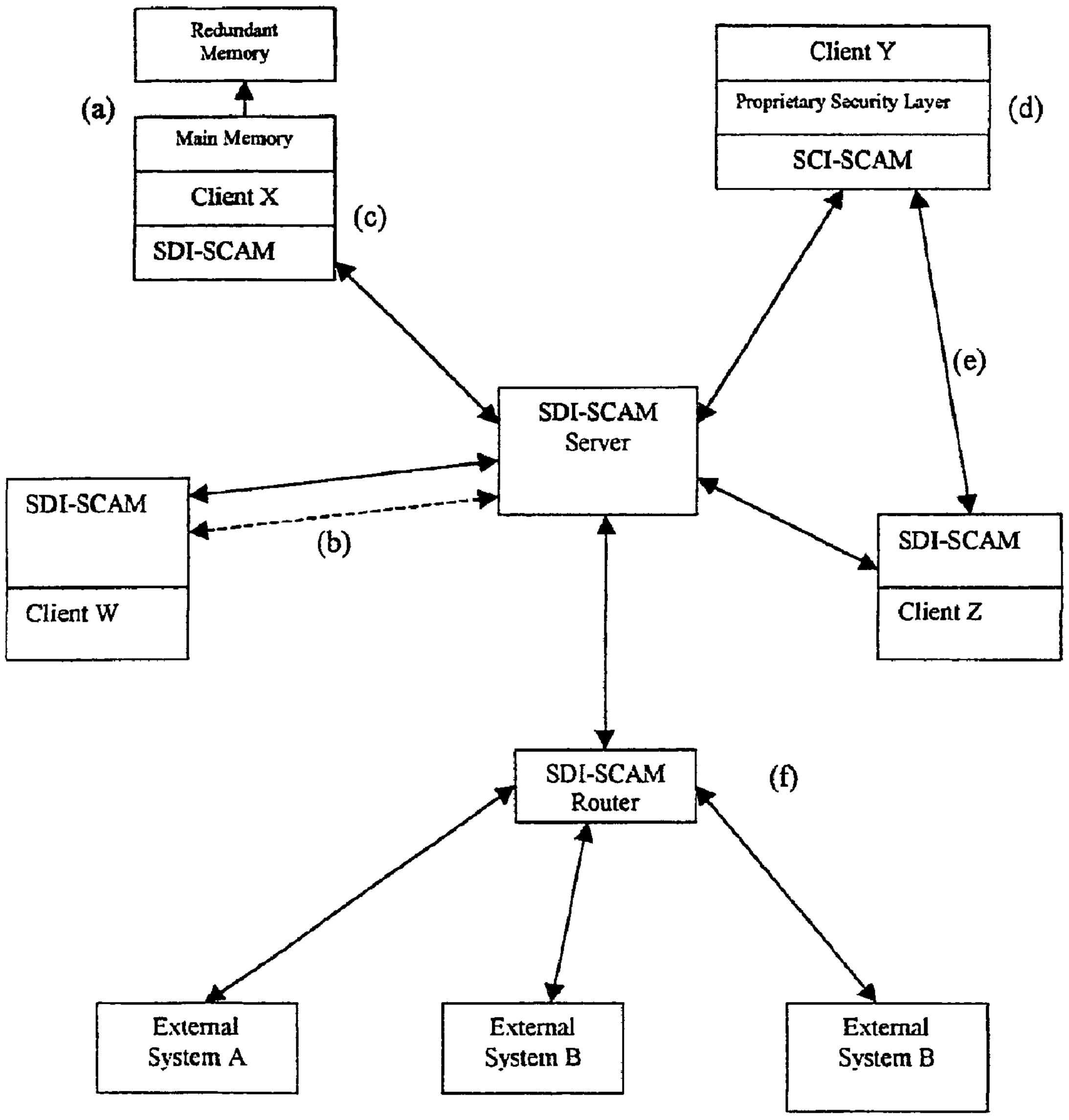
U.S. PATENT DOCUMENTS

2002/0059078	A1 *	5/2002	Valdes et al.	705/1
2002/0066034	A1 *	5/2002	Schlossberg et al.	713/201
2003/0051163	A1 *	3/2003	Bidaud	713/201
2003/0101260	A1 *	5/2003	Dacier et al.	709/224
2003/0115322	A1 *	6/2003	Moriconi et al.	709/224
2003/0145225	A1 *	7/2003	Bruton et al.	713/201
2004/0123142	A1 *	6/2004	Dubal et al.	713/201
2004/0153666	A1 *	8/2004	Sobel	713/201
2004/0165588	A1 *	8/2004	Pandya	370/389
2004/0215972	A1 *	10/2004	Sung et al.	713/201
2004/0250060	A1 *	12/2004	Diep et al.	713/155
2005/0257247	A1 *	11/2005	Moriconi et al.	726/2

OTHER PUBLICATIONS

Anderson et al., "Next Generation Intrusion Detection Expert System (NIDES) A Summary", SRI-CSL-97-07, May 1995. pp. 1-35.*

* cited by examiner


Redundant Memory
(a)
Main Memory
Client X
SDI-SCAM
(c)
Client Y
Proprietary Security Layer
SCI-SCAM
(d)
(e)
SDI-SCAM Server
SDI-SCAM
Client W
(b)
SDI-SCAM
Client Z
SDI-SCAM Router
(f)
External System A
External System B
External System B

SYSTEM AND METHOD FOR A DISTRIBUTED APPLICATION AND NETWORK SECURITY SYSTEM (SDI-SCAM)

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is a Continuation-in-Part of application Ser. No. 10/693,148, filed Oct. 23, 2003 now abandoned, and application Ser. No. 10/693,148 is incorporated herein by reference. This patent application also claims benefit of Provisional Application 60/436,363, filed Dec. 24, 2002.

BACKGROUND OF THE INVENTION

(1) Field of the Invention

The invention related to the field of security systems for computer networks.

(2) Description of Related Art

Computer networks today are as vulnerable as ever from unauthorized intrusions by external entities. The increased complexity and variety of computer systems in operation means that an even wider array of intrusive strategies is possible, in turn requiring ever more sophisticated protective mechanisms.

Although simultaneous attacks are often launched against entire networks, most existing security systems are focused at the level of the individual machine—ports are monitored for suspicious activity, incoming files are scanned for viruses, and user accounts are protected from unauthorized access. Network-level security is much harder to control—and it may take time for coordinated threats to be detected and counteracted. For example, a virus may have several days to spread and attack individual machines before public awareness of the threat emerges, and even then it may take several more days for security experts to create and disseminate a countermeasure. In the first few days of such an attack individual system operators may not realize that their systems' problems are not simply localized disturbances, but rather a network-level problem, and it is during this window of time that much of the damage is done both directly and indirectly by replication and propagation across the network(s).

BRIEF SUMMARY OF THE INVENTION

An architecture is provided for a widely distributed security system (SDI-SCAM) that protects computers at individual client locations, but which constantly pools and analyzes information gathered from machines across a network in order to quickly detect patterns consistent with intrusion or attack, singular or coordinated. When a novel method of attack has been detected, the system distributes warnings and potential countermeasures to each individual machine on the network. In a preferred implementation, such a warning may potentially consist of a probability distribution of the likelihood of an intrusion or attack as well as the relative probabilistic likelihood that such potential intrusion possesses certain characteristics or typologies or even strategic objectives in order to best recommend and/or distribute to each machine the most befitting countermeasure(s) given all presently known particular data and associated predicted probabilistic information regarding the prospective intrusion or attack. If any systems are adversely affected, methods for repairing the damage are shared and redistributed throughout the network. The net impact of SDI-SCAM is that every machine on a network can benefit from security experience gained at any other point on the network. A high and uniform level of security is therefore assured to all systems attached to the network, and this security is updated in real-time.

BRIEF DESCRIPTION OF THE DRAWINGS

The FIGURE demonstrates some of the architectural features discussed, including (a) redundant memory within a given machine, (b) redundant connections between clients and servers, (c) SDI-SCAM installed as a primary security system, (d) SDI-SCAM piggybacking on an existing security system, (e) direct client-to-client agent communications, (f) on a router.

DETAILED DESCRIPTION OF THE INVENTION

The basic architectural approach for SDI-SCAM is that each node of a computer network is loaded with an agent capable both of ensuring security at the locality of the machine on which it is installed, and of communicating with other SDI-SCAM agents across the network. Because agent configurations are highly flexible, SDI-SCAM implementations can vary widely, running the spectrum from fully centralized (in which SDI-SCAM agents on client machines communicate uniquely with a centralized server dedicated to processing security-related information) to fully distributed (in which each client agent is given the ability to process security information locally, and information is shared on a peer-to-peer basis).

Basic Network Elements

The preexisting elements of this network security system are the machines themselves. It is assumed that these systems, which act as the nodes of a network, consist of heterogeneous pieces of hardware running different sorts of operating systems. It may well be the case that various security layers will already be in place.

Additional Hardware

In preparation for the installation of SDI-SCAM across a network, it will often be desirable to upgrade existing machines with redundant hardware.

In a preferred embodiment, preexisting systems will be supplemented with redundant memory systems that persistently mirror the contents of the primary memory banks. When a computer's primary memory is corrupted (as can happen during a viral attack), it can be completed, cleared and reset with a pre-corruption image from the backup.

A further redundancy can be built into the network connections that link the local nodes to SDI-SCAM servers. For example, a computer that normally operates through land-based optic lines may be given an additional wireless connection through a satellite system.

An expensive, but preferred, architecture is to connect each SDI-SCAM agent through a fully isolated network that operates independently from the network on which the protected system resides. Thus, the SDI-SCAM agent will remain in contact with the security network even when the system it is supporting is under a sustained or unusually intense attack.

SDI-SCAM Agents

An agent is an entity that can be loaded onto any node(s) of a network, and which in this case is assigned responsibilities related to system security. Note that the construction of a given agent can vary widely, as it can be implemented through software, through hardware, through human interaction, or some combination thereof. In a preferred embodiment of SDI-SCAM, every machine linked into the system is loaded with an SDI-SCAM agent. Agent responsibilities include the following:

1) The collection of traffic data—among other things, each agent observes the packets being routed through its local system, observes every file transmission, monitors every user action, and logs every request for access.

2) The ability to communicate with other SDI-SCAM agents—each agent has the ability to communicate and exchange information with other agents (although the content of this information and the agents with which it is shared may be controlled, as will be discussed later). In normal use, a remote agent will send filtered traffic information downstream. When other agents detect potential security threats, warnings will pass upstream back to the remote agent.

3) The maintenance of various protections—On a continual basis, SDI-SCAM agents send and receive warnings and potential countermeasures relevant to whatever network risks are the most likely at a given time. For example, if a computer virus is detected at one node on the network, the local agent will immediately communicate a warning to all other agents in its contact neighborhood. If an attack is especially bad, the agent will have the ability to swap into the backup memory or contact other agents through alternative communications lines.

SDI-SCAM can operate either as a standalone security system, or as an additional layer that subsumes (and takes priority over, in cases of conflict) existing security protocols.

4) The ability to repair damage—Even after a node is known to have been attacked, the SDI-SCAM agent can be given access privileges such that it can aid the system administrator in controlling and repairing whatever damage has resulted.

5) The ability to scan collected data traffic for patterns consistent with threats—In many configurations, SDI-SCAM agents share their traffic information with a dedicated SDI-SCAM server capable of gathering and sifting through the entirety of the traffic data in order to detect patterns consistent with a network attack, be it related to a hacker or to a new virus. Certain traffic events, which individually may be mistaken as simple anomalies, may become more apparent when the totality of a network's (or multiple networks) traffic is considered on a macro scale.

6) Notifying system administrators in the event of certain probabilistic attributes exceeding certain levels—The system's implementation of a Belief network (as herein disclosed) may also be used to determine under what overall conditions of probabilistically determined and descriptive variables it is advantageous to notify the system administrator. These variables can be based upon the predicted likelihood for the system to solve the problem, prevent certain types of problems, undesirable events and/or quantified degrees thereof from occurring or manual/or manually adaptive rules may prescribe threshold settings for some or all of these key variables. Among other situations, the system administrator may be notified or alerted in cases in which patterns detected may be only slightly suspicious according to the standard screening methodology, however, are consistent with SDI-Scam's best estimated simulation model from its distributed agent sources of how a threat might emerge, e.g., by mutation and re-emergence, e.g., after initially being defeated by SDI-Scam.

Meta-data associated with the accessor like a watermark that can also be embedded in code that contains digital credentials of the user, however, incorporates the use of "potentially" rogue, irresponsible, or destructive individuals as per the types of associated predictive attributes from criteria as disclosed in a presently preferred embodiment. The code cannot be tampered with out interrupting the watermark. A more general term for this "invisible" code sequence, which

appears random to a would-be interceptor, is "embedded code". Typically, the embedding is done in a much larger nonsense message to apparently random patterns (in as much as the application code would already be encrypted) and this nonsense message content may not be required. Also, it can be associated with functionally defined portions of the code, which pre-approve certain behaviors. The system could also be based upon willingness of the accessor and/or code which s/he writes to statistically pseudonymize and profile the user with that of the patterns/types, etc. of code s/he has written in the past, thus predicting even without explicit identification who is the likely author and what s/he is like, i.e., what is the statistical probability distribution of the individual to each of a variety of previously known identities based upon code morphological characteristics, functional behavioral features, human behavioral features (e.g., if it is accompanied by a human attack). Pseudonyms and resolution credentials may be useful to authenticate the basic intent and MO of the author of the code while use of cryptographically interoperable pseudonyms, i.e., multiple unique but single identity aliases which are linkable to that single author only by SDI-SCAM for its security analytical purposes and under prescribed conditions (as data disclosure policies) as dictated by that author. Pseudonyms may be used to insure the same level of anonymity of the author as uncredentialed code. This approach could, of course, either be implemented as a local protocol (i.e., existing applications, application updates and new applications could all possess these credentials verifying/certifying that the present code was written by an individual who has been certified by a trusted certification authority as being non-malicious). This approach and the above pseudonym based identity protection scheme, while applied in this case to the application of software security are disclosed in detail for the application of identity protection, data privacy and security from rogue individuals interacting on communication networks such as the Internet. These relevantly related techniques are well described in the parent case as well as in U.S. Pat. No. 5,754,938, entitled "Pseudonymous Server for System for Customized Electronic Identification of Desirable Objects".

Within a typical context, this type of code certification should be impervious to a "man in the middle" attack. Such embedded messages (or in a similar cryptographic variation, "fingerprinting") are inherently effective for the security application proposed inasmuch as any rogue code which a system attacker would attempt to insert into a certified application or communication or other communication containing executable code would contain within its sequences continuous portions which do not contain the embedded credential-based sequences. Likewise, in case the would-be man in the middle attempted to remove certain data, (e.g., credentials or functional application code) the fingerprinting technique would recognize the specific extracted code segments. This exact same problem can be solved alternatively another way in which the primary objective is to transmit data containing a message the existence of which is not possible to be detected by a would be "man in the middle" attacker. In the example approach in which a content bearing message is embedded or fingerprinted into the application code (or less desirably in an associated larger message), the message can only be identified by the recipient (the local SDI-SCAM agent) who may also be similarly hidden or "steganographed" as with the originally sent message (in order to verify receipt of the message by the authenticated recipient. There may exist in this content bearing message a variety of useful credentials incorporated therein including but not limited to credentials approving both authenticity, untampered state and authentication of the sender and/or author as well as proof of certified "good intent" on the part of the code author. The technique for insuring that the embedded sequences are completely undetectable, while at the same time being diffusely spread throughout the code is typically performed by using encryption techniques (e.g., pseudo-random sequences) to determine the positions of the sequence bits within the remaining code in order to thus pass a message to the recipient (the local SDI-SCAM agent) containing the credentials and potentially the message of the coordinates of the associated meaningful sequences, such that all of these content bearing sequences appear among the remaining code as random noise, including the portion of the message containing the encrypted coordinate data of which coordinate bits possessing the totality of the embedded or fingerprinted message can be found within the application. Alternatively, this message containing the coordinate locations of where to find the meaningful bits containing the content bearing message may be embedded within a larger message which itself appears to consist entirely of noise (which in and of itself lends the security of the embedded or fingerprinted message contained therein). The primary hurdle in this case is to enable the recipient to be privy to certain data, which is not known to a would-be "man in the middle" attacker namely where to look for the message, i.e., the coordinates of the meaningful data constructing the message. This "shared secret" between the sender and the receiver could be conveyed to each party initially by a (one time) physical distribution (e.g., contained within an application if it is physically distributed, such as on a disk, or visa vie the OS or CPU, etc. In one variation in which the dissemination of this message needs to be performed on a network wide level (or group level), the shared secrets may be physically distributed, once to all parties in a group and, subsequently, all parties would be able to instantly initiate communications with the security guarantees achievable through the presently proposed methodology.

Finally, it will be sufficiently obvious to one skilled in the art that the presently proposed methodology has numerous potential applications in cryptography and data security and thus the means for distributing data coordinates to a recipient of a steganographed message for conveying (and if desired reciprocally confirming) a message is in no way limited to messages, containing credentials and authentication certificates about an author and/or sender. For example, the present technique could be very prudently employed as a means to distribute and replenish shared set keys within the context of the co-pending application U.S. patent application Ser. No. 10/418,983, filed Apr. 18, 2003. It may also protect against man in the middle attacks against distribution of private keys in Pki protocols.

SDI-SCAM Network

There are multiple network morphologies possible. Major configurations include the following:

1) Local network: SDI-SCAM enabled machines may form a local network, such as a LAN or WAN. Gateways to external networks (such as the Internet) can be fully controlled through SDI-SCAM enabled routers.

2) Open network: On the other hand, SDI-SCAM enabled machines can be connected directly to outside systems (such as a desktop system connecting through a generic ISP), but which maintain communications with a chosen neighborhood of other SDI-SCAM enabled machines.

3) Centrally organized networks—In this configuration, thinner SDI-SCAM agents are placed on individual nodes; these agents continue to be responsible for direct security and repair, but transmit gathered traffic information to central

SDI-SCAM servers containing dedicated hardware and software capable of swift and very in-depth analysis of the gathered information.

4) Distributed networks: In this configuration, each SDI-SCAM agent shares the responsibility for traffic data analysis and the generation of preventative measures with other agents. A peer-to-peer morphology would work well in this case.

Inter-Agent Communications

Although there is clearly a benefit for agents to fully pool all information, it may be desirable to control both the content shared and the partners with which a particular agent is allowed to interact. These parameters can be set at the local level according to users' preferences.

SDI-SCAM agents may in fact negotiate with each other depending on the value and sensitivity of particular information, as well as the value of any likely synergies between them. Multiple agents may meet in virtual information sharing marketplaces.

Another level of security can be gained through the exchange of obfuscated, but still valuable, information. Such randomized aggregates would allow systems to share fundamentals without revealing details of their particular data (for example, agents could share times of attempted log-ins without revealing the associated user ids and failed passwords).

In more complex realizations of this system, associated groups of agents may form coalitions, with information shared freely internally, but shared with conditions externally.

A further feature is that communications between agents need not be perfectly symmetric—in other words, different agents may send and receive different sorts of information. This might apply, for example, to a centrally organized SDI-SCAM network: outlying agents would have no need to transmit detailed traffic data to each other, but would rather transmit it directly to a central server. The central server might communicate with other central servers, in which case it would transmit high-level information relevant to the processing of the entirety of the traffic data; on the other hand, when communicating with outlying nodes, the central server might only transmit simple virus protection instructions and metrics which are substantially devoid of any data which suggests what types of information, attacker strategies or applications are running on other nodes on the system which are outside of the network of nodes and which are currently trusted by the nodes from which the centrally collected and processed data had been acquired.

Furthermore, there may be an additional or alternative approach to guaranteeing absolute data security at a local network or machine level while enabling maximal or complete harnessing of all of the statistical knowledge, which is present across the entirety of the network. In this approach it may be possible to operate SDI-SCAM or certain particularly sensitive portions of it with its multiple agent architecture as a singular trusted, yet distributed multi-agent system. In this variation, all of the locally performed or assigned agent functions are assumed to contain sensitive data belonging to external third parties and thus all processing activities, data communications with other agents or the central SDI-SCAM server occurs within a secure trusted and untamperable environment such that the only knowledge ultimately accessible by any given agents, associated local server or network on which it physically resides may be the collection of executed functions which are performed by the local agent on behalf of the SDI-SCAM to protect the local system as herein disclosed.

The order and way in which agents communicate with each other may be highly conditioned on the particular nature of a given system. Criteria include (but are not limited to) the following:

overall vulnerability of a system;

importance of the system to the integrity or functioning of a network;

sensitivity and value of the data stored on a system;

probability that the system has already been compromised or damaged;

characteristics of the network traffic going to and coming from the system;

overall importance of a system to a potential or identified hacker or specific system subcomponent.

This may dynamically change from moment to moment and is predicated by a probabilistic estimate determination variable of the intruder, whether autonomous or human and/or by human expert based estimates who are ideally familiar with local competition (or enemies) and broad knowledge of what types of knowledge on the system would be most of interest to which other entities or individuals and for what reason. If an individual is specifically identified this statistical model may further borrow and integrate techniques disclosed in co-pending U.S. patent application Ser. No. 10/202,302, filed Jul. 24, 2002.

Updates and communications between agents (termed "polling") may be based on schedules or on circumstances. For example, a remote agent may be updated with new anti-viral software once a month; however, if any other node on the network is attacked, the schedule is suspended and an immediate update is performed. Certainly even if an attack which, for example, has only begun to occur or which has not even positively been confirmed as yet, triggers SDI-SCAM's system alert feature, other nodes on the network most preferentially/urgently those which are physically proximal or in other ways similar may also be put on alert status and SDI-SCAM's repertoire of protective features may be triggered so as to begin operating at a heightened level of defensive activity. As indicated, there may be a range of different system defense levels corresponding to a decreased probabilistic likelihood of a threat and the likely severity thereof should this threat exist. Local system administrators are notified appropriately as well. Determining the likelihood that a threat upon a particular node or network will also be carried out against any other given node can be predicted by such variables as commonalities at an organizational or strategic level, data communication occurring there between, commonalities in the existing or perceived data on applications contained or functional objectives achieved upon that node, presume interest level that a potential intruder of the attacked node or network may also have with the other node, etc.

Polling priority may be based on calculated likelihoods: for example, if various factors indicate that the probability is high that a remote node has been infected by a particular type of virus, the central server may be put into immediate communication. Polling priority will also depend on the nature of the nodes and the way in which their agents have been seen to communicate. U.S. Pat. No. 5,754,939, entitled "System for Generation of User Profiles for a System for Customized Electronic Identification of Desirables Objects" may be used as the basis for optimizing the way in which polling is performed.

Illustration

FIG. **1** provides an illustration of some of the configurations discussed here.

Analytics

Given the number of different security objectives, as well as the number and diversity of possible agents and network configurations, a fairly broad range of analytical tools are employed by SDI-SCAM. They include, but are not limited to, the following major categories of analysis:

Methods to Detect and Classify Direct Intrusions

Direct intrusions are attempts by unauthorized entities to enter a particular system, either over a network or through local terminals. These can range from fairly unsophisticated attacks (for example, teenage "script kiddies" using standard public domain software to scan for open ports across a list of target IP addresses), to extremely skillful attacks that are focused on a very particular target (as might happen during corporate espionage).

Since SDI-SCAM agents are able to dynamically monitor and analyze as well as control all in-going and out-going traffic, they are in a good position to detect and counteract such attacks.

1) Attack Patterns Consistent with Previously-Observed Patterns Across the SDI-SCAM Distributed System.

Each SDI-SCAM agent has access to a shared database that contains the signature patterns of previously observed (as well as verified) attacks. The likelihood of these events having been actual attacks may be probabilistically estimated so as to optimize the precision of SDI-SCAM detection/diagnosis as well as countermeasure deployment system modules. Such patterns might include the use of a particular password list, log-ins at particular time intervals or frequencies or times, log-ins from suspect IPs, (and/or combinations thereof) constitute a few of the straightforward examples.

If such a pattern is detected, the resident SDI-SCAM agent may opt to deny all entry to the IP of the incoming log attempts, or it may opt for a more sophisticated defense, such as opening a "honey pot" trap, a virtual space that simulates the environment of the system that is being protected. The hacker, believing that he has actually broken into the system, can then be monitored by SDI-SCAM, as his behavior might gives clues to his location, identity, and motives and incriminatory evidence, if desired. Assuming the hacker has learned (or possesses) enough knowledge about the system to detect "honey pot" traps it is advantageous and precocious to possess at least equivalent knowledge regarding SDI-SCAM to possess at least equivalent knowledge regarding its own environment and to be able to enable the system administrator access to that knowledge as well as (via SDI-SCAM) knowledge known or suspected to exist within a probabilistic context regarding the hacker or threat and its strategy and/or this knowledge may be acted upon appropriately by SDI-SCAM in automatic mode. Invariably all counter measures (such as honey pot traps) used by SDI-SCAM can be used to the advantage of the hacker if s/he is aware of the strategy of SDI-SCAM to monitor, model, locate in order to ultimately catch him/her.

2) Utilizing Data Modeling to Adaptively Learn and Recommend Appropriate Countermeasures

Implementation of practically viable automated countermeasure scrutinization and recommendation scheme is quite achievable:

a. If the conditions/parameter triggers are simple and unambiguous, and

b. If the system administrator is notified and able to intervene while exploiting the system's analytical knowledge and system-generated recommendations and scrutinies by the system on behalf of his/her chosen response decision.

In the ideal scenario, because rogue attacks are capable of performing increasingly effectively against system security protections (in addition to being more sophisticated and expeditious) and especially with regards to leveraging the system's own abundantly capable resources, it may be ideal as a complementary measure to building redundancy into the system resources in the interest of expediency of decrypting a counter measure, to also immediately respond in automatic mode, then solicit the active, albeit system-guided intervention of the system administrator whereby more significant decisions can be perhaps more confidently and prudently executed (e.g., whether or not to delete potentially corrupted files/portions of system data at the server or network level), whether to guarantee a certain portion of the network but allow certain essential functions to continue for the time being without code exchange, whether or not to attempt to infect the hacker's machine (or analysis code into the virus itself) which may provide additional detailed information as well, etc.

3) Novel Attacks

In some cases, attacks will follow completely new or novel patterns.

Such attacks can be detected in different ways. One solution is to configure a Bayesian network to constantly gauge the probability of an ongoing attack by monitoring network traffic activity (this configuration can be done by human experts and/or through machine learning techniques). A variety of factors can be extracted from the network traffic across all SDI-SCAM agents in the local network—for example, the number of failed log-ins, the identities and IP addresses of those users attempting to log in, the importance, sensitivity or "value" (more specifically "perceived value") of particular target files or contents potential adversarial entity or prospective hacker, etc. These factors are fed into ongoing probability calculations, which may trigger a system-wide warning if a certain threshold is surpassed.

Keystroke monitoring virus must be mentioned since it is impervious to NORTONT™, etc.

For example, suppose a ring of corporate spies tries to hit a company's network simultaneously. SDI-SCAM agents across the network will report the use of unauthorized passwords originating from the same IP or IPs to which associations have been constructed via SDI-SCAM based upon historical statistics if the probabilistic likelihood of such events occurring independently might be so unlikely that the Bayesian network would immediately increase its estimate of an ongoing attack.

4) Attack Warnings

Note that in all cases, when an attack is suspected the resident SDI-SCAM agent will immediately alert all the other SDI-SCAM agents in its network neighborhood, sharing all traffic information relevant to the on-going security issue. Such warnings will include information related to the particular nature of the problem, in particular the probability and nature of the threat (for example, communication with an unsecure system, access by an authorized user, reception of potentially infected files, etc.).

When an on-going attack is announced, SDI-SCAM agents receiving this information may opt to increase the security levels of their own systems. For example, users may be required to telephone at the time of their log-in to verify location (through caller ID) and voiceprint.

Methods to Detect and Classify Viruses or "Trojan Horses"

Origins, possible paths of transmission across sites, etc. types of files (e.g., particularly vulnerable or vulnerable origin site), may be analyzed to provide ideas as to how to use this data to make a vulnerable application, Trojan horse attempt impervious, make rogueness, crypto query, even rewrite code.

Another vector of attack is through viruses (which are often unauthorized and malicious programs attached to files, email, or documents) and trojan horses (seemingly innocuous programs that contain hidden programming capable of causing damage).

Code Analysis

The conventional viral detection methodology is to scan the code (in the case of executable modules) and macros (in the case of smart documents, such as those generated by Microsoft WORD™) for patterns that have previously been associated with viruses or malicious programming.

SDI-SCAM maintains up-to-date records of all known viruses and checks all incoming files (and periodically, all stored files) against these records. A match indicates that a file is potentially infected—the user is alerted of the danger and automatic defensive measures may be set into motion.

Behavioral Analysis

SDI-SCAM monitors all processes for behavior consistent with viral infection. For example, a program that is observed to open and modify a wide range of heterogeneous files, which accesses the mail system's address folder, which aggressively propagates copies of itself, which engages in recursively redundant actions whose objective is designed to achieve no useful purposes or frequently which aggressively/repetitively generates or obtains data files in order to propagate inordinately voluminous and/or large files (possibly including itself) resulting in bursts of traffic (thus overloading valuable network transmission capacity), which performs similar recursively redundant actions resulting in consumption and overloading of valuable processing capacity, which modifies or mutates its own code (and/or behavior), or which opens unexpected communication ports with outside entities will be flagged as a potential threat. Unquestionably, SDI-SCAM's highly distributed data traffic monitoring and behavior and code analysis facilities as a combined approach give it a marked and compelling advantage in rapidly analyzing those behavioral patterns and characteristics most commonly associated with a rogue code such as viruses, Trojan horses, worms, etc. whose tell tale signs could not be identified nearly as expeditiously as that of SDI-SCAM's distributed agent monitoring architecture. Such commonly occurring signatures which SDI-SCAM's distributed Bayesian methodology is particularly well suited includes those patterns of self replication and dissemination through address books, email, web browsing sessions, etc., as well as the co-occurrence of identical or related patterns of behavior and code sequences in conjunction with these replicating and self propagating patterns as observed only on a network level. Certainly part of this behavioral analysis may encompass attempts by SDI-SCAM to classify the identity or type of virus based upon all of the above observed characteristics as well as attempting to extrapolate its high level objectives and associated executable rule sets based upon its behavioral patterns associated with the conditions/variables of the environment which it has encountered, the data which it has likely accessed, the actions, events and countermeasures to which it has been exposed, the code within which it has likely been embedded, etc.

Although it may be difficult to delineate rogue from innocuous code it is certainly within the scope of capabilities of SDI-SCAM to utilize all of the available data, both behavioral and code sequences, in order to attempt to reverse engineer the code for the purposes of both predicting its future behavior, likely past behavior and high level objectives. For example, SDI-SCAM could replicate the code inside of an environment which is quarantined from the network, but which is a replica of the network or a portion thereof. SCI-SCAM could then monitor how the code behaves in this simulated environment to the actual one as well as observing its response to targeted stimuli, which may, for example, provide opportune conditions for the most likely rogue actions to be performed. This analytical procedure may be performed in response to a predicting statistical model (designed to predict the code's behavior) when a decision tree could be used to dynamically select the set of functions to be executed which based upon the same model are correlated and then predicted to elucidate responses on which are the most optimally revealing, reveal the most revealing which is needed to complete the construction of this data model for the codes for being able to predict the code's behavior across a wide array of conditions, actions, software and data to which it may ultimately become exposed within the entirety of network(s). In depth analysis of potentially suspicious code although challenging as it may be could potentially provide system level insights into how to best respond to the potential threat and if mandatory the nature and aggressiveness of countermeasures to be taken or recommended to the appropriate human system security counterpart.

The user will be alerted, and if he confirms that the program is operating outside of expected parameters, or if the user does not recognize the program, it is taken offline until it can be examined in detail by an expert.

Dead-Ringer Analysis

Although not currently a threat, it is likely that infectious programs will be able to simulate the behavior of human users. A suite of behavioral response tests can be developed to detect and counteract such entities, e.g., a probabilistic model based upon other previous threats in the statistically similar characteristics (including behavioral characteristics and certainly those determined to be the most likely to be the same). Queries which may be required of the "user" to be answered correctly or to perform a task (e.g., compose a block of text on the basis of a query) in order to proceed could be solicited of the user which are crafted such that an emulating virus would likely fail such query procedure. Moreover, Natural Language Processing methods can be used to analyze outgoing text for irregularities consistent with a non-human origin. It is possible that in a similar fashion, that, in theory very smart emulations of existing code could be manually or even automatically on the fly created which emulates in many respects existing "good code", but which actually is designed for malicious objectives or, for example to take over control of the good code or replace it with the rogue version. As additional attributes of the system, the system may determine probability and degree of ill motive of individuals of most likely suspicion (if such suspicion is high enough to be of reasonable concern). Typically, common suspicion of particular individuals can be linked to unscrupulous employees (present or former), disgruntled employees, disgruntled spouses of key persons/owners (e.g., changing files, information release, etc.) to embarrass or defame the person or to feign a verbal or tactical attack on a friend, associate or colleague. Such "suspects" could also include trusted partners who may be confided with knowledge of the existence of unique information which could be of interest directly or could even help or strengthen that party in its business position with its "trusted" business partner.

Control of Triggers

If the probability of an infection is deemed to be high, SDI-SCAM may control the generation of events that could

potentially trigger the reaction of a resident virus. For example, if a bank suspects that a corporate virus has infected the system, all transactions may be suspended until the virus is cleared. Otherwise, the action of a user requesting an online transaction (thereby releasing his personal password to the system) may trigger the virus into capturing and re-transmitting his personal information.

Tracing Threats Back to their Original Source

In traditional system security techniques this objective is highly desirable and yet extremely difficult. Nonetheless, SDI-SCAM's functional features lend themselves quite well to the design of certain particular types of applications, which can be useful in addressing this particular problem. For example, the following example applications may be herein considered:

1. "Infecting" the hacker's machine (or the virus) with a virus, which logs and/or conveys back to the SDI-SCAM agent the location, behavior, files infected as well as all IP addresses of the machines in which these files reside. This approach is likely to work provided that the implanted virus by SDI-SCAM is not recognized by standard virus scanning software or other IDS systems and assuming that the receiving machine is not programmed to block any outgoing messages. Thus, the success would be determined in part by the effectiveness of the virus to take control of the adversary's (or rogue virus containing) machine. This type of direct analysis will both enable preemptive alerts of exactly where the virus may be spreading to other machines and/or networks as well as provide valuable statistically confident data as to the function, behavior, data or code affinities and behavior in response to infection of the same as well as epidemiological characteristics which could be extremely valuable as to anticipatory determination and qualification of the associated threat on other machines, as well as the most appropriate countermeasure each local agent should implement or receive in response. Certainly, this approach could be useful for viruses, which possess particular rapidly proliferating characteristics, rapid infliction of destructive behavior. For example, one could imagine the behavior of more sophisticated viruses which might proliferate themselves as redundant messages so as to rapidly overwhelm network capacity and/or memory processing and/or implement parallel strategies.

This approach could also enable SDI-SCAM to model not only future epidemiological characteristics of rogue software but also that of post epidemiological behavior (which machines or networks were likely to have been infective previously based upon presently known epidemiological characteristics) and the devices/networks which are known to be and probabilistically suspected of being infected by the same virus (or mutated variant thereof). Certainly reconstruction past, present and future behavior in this regard could be relatively easy to perform for worms that may require access to ISP server logs for other variations which may use email and web server connections as a medium of transmission. A protocol also may allow for the existence of a latent tracking virus to reside within all machines which can be, in the case of significant probability of a threat in and among a network community or otherwise "group" an excessive probability of a threat, the tracking virus may be remotely activated by a multi-casted activation message originating form a core (or root) server.

2. Use of SDI-SCAM Architecture for Application Level Security

It will be increasingly important in the future for many of the functions of SDI-SCAM as implemented within the context of its presently disclosed distributed statistical analytics to be implemented not only at the level of a distributed network security system but also at the individual application level. That is to say that SDI-SCAM agents could, in addition to the above described system level implementations, also implement their various functions for data collection, analysis, and countermeasures at the application level as well both to implement other application level security protocols as well as incorporate into the statistical analytical scheme probabilistic attributes regarding the behavior functions, etc., of such rogue code within the context of the particular relevant applications in need of protection, albeit using the same distributed adaptive modeling and countermeasure response protocols described herein in comprehensive fashion.

Methods to Detect Tampered Files (Semantics and Content)

It is sometimes the case that intruders, rather than destroying or removing files, will simply alter them in potentially malicious ways. For example, students may attempt to hack into their school system in order to change grades, or a more advanced hacker may attempt to break into a bank to fraudulently increase the balance in his account, into tax or criminal record databases in order to change tax liabilities, records of property ownership or criminal records, into professional board's databases in order to change licensure status. Similar tampering may occur to files whose contents may relate to the hacker (e.g., employee files of present or past employers). Malicious code may, in theory, perform all of the functions that a human may perform, perhaps, however, potentially even more unobtrusively and elusively in that it may be more difficult to trace and flag than a human if the code is very small, robust and capable of focused but sophisticated emulations of legitimate applications and users.

In addition to the above suggested techniques for use in tampering detection and ultimately prevention (or even tracing the origins of tampering attempts), there are other straightforward IDS-based approaches by which such attempts could be countered (and could even complement the above safeguarding scheme, for example, in terms of being a default detection scheme and/or in corroboration of the presumed integrity of credentialed individuals). Thus, the following IDS-based alternative technical approach is also provided as well. The local SDI-SCAM agent maintains logs that detail the general characteristics (size, word count, hash code) of all documents on the system. The time and circumstances of any changes are cross-checked against average traffic patterns for that particular system. Hence, school records altered at 3 am (in a district where all school secretaries worked strictly from 9 am to 5 pm) may be flagged as potential objects of tampering.

Tampered files will sometimes show a marked change in writing style or technique. Natural Language Programming (NLP) techniques may be used to detect such changes. Certainly in the event of these suspicious activities and other conditions, it may be advantageous to retain not only the associated statistical data (as the SDI-SCAM does automatically) but also details regarding the events. This could, for example, be later analyzed by humans to compare with other similar suspicious patterns also captured in detail in order to attempt to identify patterns, more subjective signatures, or hall marks which may not be able to be performed automatically (such data may also be useful for potential legal evidence).

Methods to Detect and Classify Untruthful Commercial Messages

Untruthful messages represent a more traditional kind of deception—the technology of the delivery is not damaging, rather, the content of the message itself is untruthful and may prove harmful if taken at face value by the receiver. A good

example of this is the "Nigerian Scam," a widely disseminated email that purports to be authentic, asking the receiver to give the sender access to an American bank account in exchange for great financial reward. The result, of course, is that the receiver ends up being defrauded of large amounts of money.

1) Cross-Checking Content Against Known Hoax Documents

SDI-SCAM maintains a database of questionable messages and uses natural language programming-based techniques to compare incoming messages with previously logged deceptions. Thus, when a suspicious message is detected, the receiver may be sent a secure attachment by SDI-SCAM with an email stating that there is a high probability that the mail is untruthful, and indicating pointers to web pages that discuss that particular deception. If a user is nonetheless deceived by such a message, the local SDI-SCAM agent may be alerted. It will transmit the text of the novel message to a security database, allowing every other SDI-SCAM in that network neighborhood to be aware of the danger. In such a case, the agents may retroactively warn users by scanning old emails and alerting receivers of possible deception.

Certainly in such an event, autonomously implemented counter measures may also be performed if appropriate as a defensive or evasive action or deterrent, e.g., if a pass code was inadvertently sent out (and it was not blocked by the system) the pass code could be automatically changed or temporarily frozen or if a personal bank account or credit card number were sent out in a suspected inappropriate context (again assuming it was not blocked at the source by the system), the account could be automatically temporarily frozen and the number changed or (for example) the account automatically set up as a honey pot trap to acquire just enough information about who the suspect entity is in order to catch him in an inappropriate act of fraud or deception.

2) Predicting Possible Hoax in Novel Message

In cases where a message is not closely correlated with known hoaxes, it is still possible to analyze (using natural language processing techniques that are currently well known to the art) the content of the message and flag any suspicious content:

the content of the message can be cross-checked against recent news stories discussing hoaxes; and.

the names and return email addresses of the incoming mail may be checked against those of known hoaxsters.

Automated semantic analysis of the message may be performed for language consistent with persuasion or appeal to greed (or other weaknesses). This analysis is performed on the basis of adaptive rules which may be updated with feedback.

The identity and personal profile of the receiver may be correlated with the characteristics of known victim groups. For example, messages sent to rich elderly individuals may be given additional scrutiny.

The purported identity of the sender can be checked against the path of the email. For example, a message claiming to be from the IRS should trace back to an official government system.

A probabilistic assessment of the likelihood that the sender is fraudulent may be performed through a modified version of the system described in co-pending U.S. patent application Ser. No. 10/202,302 in which the system's probabilistic determination of predictive attributes relevant to an association with fraudulent, unscrupulous or disruptive behavior (in an on-line context) is performed—of course, the sender if self identified may also be fraudulent. The on-line sender just prior to the first receiving node on the system may also be analyzed which is a reasonably reliable tracking means if SDI-SCAM is a ubiquitous protocol (e.g., for patterns of being the origination node for previous problematic messages and/or the techniques disclosed in the same co-pending patent application), whereby the system may probabilistically predict the suspicion level of an individual(s) or organization(s) associated with that sender as being linked to other scams and/or other illegitimate or questionable activities. Related techniques may use other advanced customized semantic analysis and/or adaptive rule based/statistical techniques (preferably in combination) in order to estimate the degree of potential harmfulness of the content.

The content may be corroborated with the content of known and trusted documents, e.g., through the use of content matching techniques. More elaborate extensions of this approach may include more advanced semantic analyses of the subject content with its credible and updated/current matching counterparts whose algorithms are custom configured to confirm (or alternatively flag) or assess the probabilistically estimated "truthfulness" of contents (where "truthfulness" may be reassured according to "confirmed with credible source" as well as scalar measures of degree of likelihood of untruthfulness if the source is unconfirmed or, for example, exhibits semantically suspicious inconsistencies with itself, with credible sources or other patterns which are consistent with fraudulent or deceptive material).

The system may also detect suspicious content, for example, if its appearance co-occurs in the same message with rogue code (for example) is co-located (in the same portion of content) as a macrovirus.

Methods to Repair Post-Attack Damage

In some cases, despite the security, a system in an SDI-SCAM network may be damaged by an attack. If the attack is completely novel, a human expert may be called in to fully analyze the situation and develop appropriate repair protocols. These can then be sent to a central SDI-SCAM damage-control database for use in future situations. In this way capturing as much data and statistical information regarding the attack and its historical counterpart is valuable both as analysis data for the human or to enable the system to construct its own optimal repair protocol.

If an attack method is not novel, the local SDI-SCAM system may access this same damage repair database for solutions to the local problem. Among the remedies to damage from an attack: users are alerted, suspicious files are deleted, backup files are loaded, and current memory is completely cleared and reloaded with an image from a pre-attack state.

The invention claimed is:

1. A distributed security system that protects individual computers in a computer network having a plurality of computers, said system comprising individual computers having agents associated therewith that control the associated individual computer, each agent performing the steps of:

creating statistical models of usage of the associated individual computer in said computer network;

gathering and analyzing information relating to current usage of the associated individual computer in said computer network;

determining from said information a pattern of usage of the associated individual computer that is consistent with intrusion or attack of the associated individual computer or the computer network;

determining a probability of the likelihood of an intrusion or attack from said pattern of usage of the associated individual computer;

distributing in real-time warnings and potential countermeasures to agents of each of said individual computers in said computer network when the determined probability of the likelihood of an intrusion or attack exceeds a statistical threshold, wherein at least one of said warnings comprises information related to the nature of the intrusion or attack and the determined probability of the likelihood of intrusion or attack based on the statistical models of the associated individual computer; and

updating said statistical models of the associated individual computer to reflect the current usage of the associated individual computer in said computer network and the likelihood of intrusion or attack;

wherein each said agent schedules the associated individual computers for different anti-viral software updates based on different levels of probability of an intrusion or attack for each individual computer based on the statistical model for each individual computer and a detected level of probability of an intrusion or attack; and

wherein each said agent suspends said schedule and immediately provides the anti-viral software update to the associated individual computer when an intrusion or attack of any computer in said computer network is detected or the detected probability of an intrusion or attack is high that the associated individual computer has been infected by a particular type of virus.

2. The system of claim **1**, wherein said at least one warning comprises a relative probabilistic likelihood that a detected potential intrusion or attack based on said detected pattern of usage has certain characteristics or strategic objectives and said potential countermeasures comprise a most appropriate countermeasure for one or more most probable intrusions or attacks based on said detected pattern of usage.

3. The system of claim **1**, wherein at least one of said agents distributes to said individual computers in said computer network information relating to repairs conducted in response to detected intrusions or attacks.

4. The system of claim **1**, wherein each said agent is provided in a security system installed in each of a plurality of individual computers in said computer network.

5. The system of claim **1**, wherein each said agent is provided in a system that is provided in addition to a security system installed in each of a plurality of individual computers in said computer network.

6. The system of claim **1**, wherein each said agent is provided on a router for providing said information to individual computers or computer networks connected to said router.

7. The system of claim **1**, wherein agents associated with individual computers in said computer network communicate said information and said real-time warnings and potential countermeasures in a peer to peer fashion.

8. The system of claim **1**, wherein agents associated with individual computers in said computer network communicate said information and said real-time warnings and potential countermeasures to a central server for distribution to agents associated with other individual computers in said computer network.

9. The system of claim **1**, wherein individual computers in said computer network each comprise redundant memory systems that persistently mirror the contents of primary memory of each said individual computer.

10. The system of claim **1**, wherein said information comprises at least one of traffic data amongst individual computers in said computer network and metadata associated with one or more accessors of said individual computers in said computer network.

11. The system of claim **1**, wherein each said agent comprises a virus scanner that checks code of each said individual computer for patterns associated with viruses or malicious programming.

12. The system of claim **1**, wherein each said agent distributes shared set keys and parameters for updating said statistical models to each of said individual computers in said computer network.

13. The system of claim **1**, wherein each said agent communicates with other individual computers based on characteristics of the computer network that may change from moment to moment based on a probabilistic determination of a threat level of a particular intruder or attack.

14. The system of claim **1**, wherein each said agent schedules said different individual computers for different anti-viral software updates based on different levels of probability of an intrusion or attack for each different individual computer.

15. The system of claim **1**, wherein each said agent comprises a Bayesian network that constantly gauges a probability of an ongoing attack based on traffic activity in said computer network.

16. The system of claim **1**, wherein said warnings comprise information related to a particular nature and probability of a detected intrusion or attack.

17. The system of claim **1**, wherein said detected pattern of usage is compared against a security database to determine if said pattern of usage is consistent with a known intrusion or attack.

18. A method of protecting individual computers in a computer network having a plurality of computers, wherein individual computers have associated agents that control said individual computers to perform the steps of:

creating statistical models of usage of said individual computer in said computer network;

gathering and analyzing information relating to current usage of said individual computer in said computer network;

determining from said information a pattern of usage of said individual computer that is consistent with intrusion or attack of the individual computer or the computer network;

determining a probability of the likelihood of an intrusion or attack from said pattern of usage of said individual computer;

distributing in real-time warnings and potential countermeasures to agents of each of said individual computers in said computer network when the determined probability of the likelihood of an intrusion or attack exceeds a statistical threshold, wherein at least one of said warnings comprises information related to the nature of the intrusion or attack and the determined probability of the likelihood of intrusion or attack based on the statistical models of said individual computer;

updating said statistical models of said individual computer to reflect the current usage of said individual computer in said computer network and the likelihood of intrusion or attack;

scheduling the individual computers associated with respective agents for different anti-viral software updates based on different levels of probability of an intrusion or attack for each different individual computer based on the statistical model for the computer to be updated and a detected level of probability of an intrusion or attack; and

suspending said schedule and immediately providing the anti-viral software update to the individual computer associated with an agent when an intrusion or attack of any computer in said computer network is detected or the detected probability of an intrusion or attack is high that the computer associated with the agent has been infected by a particular type of virus.

19. The method of claim **18**, wherein said warning comprises a relative probabilistic likelihood that a detected potential intrusion or attack based on said detected pattern of usage has certain characteristics or strategic objectives and said potential countermeasures comprise a most appropriate countermeasure for one or more most probable intrusions or attacks based on said detected pattern of usage.

20. The method of claim **18**, wherein at least one of said agents further distributes to said individual computers in said computer network information relating to repairs conducted in response to detected intrusions or attacks.

21. The method of claim **18**, wherein said information and said real-time warnings and potential countermeasures are communicated amongst agents of individual computers in said computer network in a peer to peer fashion.

22. The method of claim **18**, wherein said information and said real-time warnings and potential countermeasures are communicated from agents of individual computers in said computer network to a central server for distribution to other individual computers in said computer network.

23. The method of claim **18**, wherein said information comprises at least one of traffic data amongst individual computers in said computer network and metadata associated with one or more accessors of said individual computers in said computer network.

24. The method of claim **18**, wherein each said agent further checks code of each said individual computer for patterns associated with viruses or malicious programming.

25. The method of claim **18**, wherein each said agent further distributes shared set keys and parameters for updating said statistical models to each of said individual computers in said computer network.

26. The method of claim **18**, further comprising an individual computer communicating with other individual computers based on characteristics of the computer network that may change from moment to moment based on a probabilistic determination of a threat level of a particular intruder or attack.

27. The method of claim **18**, wherein each said agent further schedules said different individual computers for different anti-viral software updates based on different levels of probability of an intrusion or attack for each different individual computer.

28. The method of claim **18**, wherein said warnings comprise information related to a particular nature and probability of a detected intrusion or attack.

29. The method of claim **18**, wherein said detected pattern of usage is compared against a security database by at least one of said agents to determine if said pattern of usage is consistent with a known intrusion or attack.

30. A distributed security system that protects individual computers in a computer network having a plurality of computers, wherein said distributed security system comprises at least one security computer that is fully isolated from said computer network having said individual computers and said at least one security computer includes an agent that controls said at least one security computer to perform the steps of:

creating statistical models of usage of said individual computers in said computer network;

gathering and analyzing information relating to current usage of said individual computers in said computer network;

determining from said information a pattern of usage of said individual computers that is consistent with intrusion or attack of the individual computers or the computer network;

determining a probability of the likelihood of an intrusion or attack from said pattern of usage of said individual computers;

distributing in real-time warnings and potential countermeasures to agents of each of said individual computers in said computer network when the determined probability of the likelihood of an intrusion or attack exceeds a statistical threshold, wherein at least one of said warnings comprises information related to the nature of the intrusion or attack and the determined probability of the likelihood of intrusion or attack based on the statistical models of said individual computers; and

updating said statistical models of said individual computers to reflect the current usage of said individual computers in said computer network and the likelihood of intrusion or attack;

wherein said agent controls said at least one security computer to perform the steps of:

scheduling different individual computers for an anti-viral software update based on the statistical models of the individual computers and a detected level of probability of an intrusion or attack of the individual computers; and

suspending said schedule and immediately providing the anti-viral software update to at least one individual computer when an intrusion or attack of any computer in said computer network is detected or the detected probability of an intrusion or attack is high that said at least one individual computer has been infected by a particular type of virus.

31. The system of claim **1**, each agent further receiving warnings and potential countermeasures from agents of other of said individual computers in said computer network when the determined probability of the likelihood of an intrusion or attack exceeds a statistical threshold, wherein at least one of said warnings comprises information related to the nature of the intrusion or attack and the determined probability of the likelihood of intrusion or attack based on the statistical models of said individual computers.

32. The method of claim **18**, each agent further receiving warnings and potential countermeasures from agents of other of said individual computers in said computer network when the determined probability of the likelihood of a intrusion or attack exceeds a statistical threshold, wherein at least one of said warnings comprises information related to the nature of the intrusion or attack and the determined probability of the likelihood of intrusion or attack based on the statistical models of said individual computers.

33. The system of claim **30**, each agent further receiving warnings and potential countermeasures from other of said individual computers in said computer network when the determined probability of the likelihood of an intrusion or attack exceeds a statistical threshold, wherein at least one of said warnings comprises information related to the nature of the intrusion or attack and the determined probability of the likelihood of intrusion or attack based on the statistical models of said individual computers.

* * * * *

Exhibit B

to

Complaint

for Patent Infringement

The ’614 Patent

US009438614B2

(12) **United States Patent**
Herz

(10) **Patent No.: US 9,438,614 B2**
(45) **Date of Patent: *Sep. 6, 2016**

(54) **SDI-SCAM**

(71) Applicant: **Fred Herz Patents, LLC**, Milton, WV (US)

(72) Inventor: **Frederick S. M. Herz**, Milton, WV (US)

(73) Assignee: **Fred Herz Patents, LLC**, Milton, WV (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/942,175**

(22) Filed: **Jul. 15, 2013**

(65) **Prior Publication Data**

US 2013/0305377 A1 Nov. 14, 2013

Related U.S. Application Data

(63) Continuation of application No. 13/279,893, filed on Oct. 24, 2011, now Pat. No. 8,490,197, which is a continuation of application No. 10/693,149, filed on Oct. 23, 2003, now Pat. No. 8,046,835.

(60) Provisional application No. 60/420,754, filed on Oct. 23, 2002.

(51) **Int. Cl.**

H04L 29/06	(2006.01)
G06Q 20/20	(2012.01)
H04L 12/24	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 63/1433*** (2013.01); ***G06Q 20/201*** (2013.01); ***H04L 41/147*** (2013.01); ***H04L 41/16*** (2013.01); ***H04L 63/145*** (2013.01); *H04L 41/0853* (2013.01)

(58) **Field of Classification Search**

CPC G06Q 20/201; H04L 41/0853; H04L 63/1433; H04L 41/16; H04L 63/145; H04L 41/147

USPC 726/25

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,754,939 A		5/1998	Herz et al.	
6,405,250 B1	*	6/2002	Lin et al.	709/224

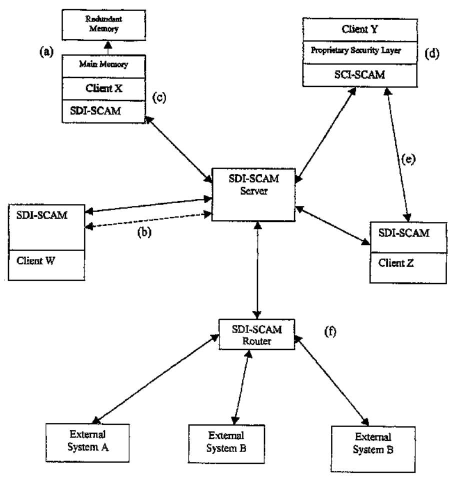
(Continued)

Primary Examiner — Kambiz Zand
Assistant Examiner — Aubrey Wyszynski
(74) *Attorney, Agent, or Firm* — Baker & Hostetler LLP

(57) **ABSTRACT**

A distributed multi-agent system and method is implemented and employed across at least one intranet for purposes of real time collection, monitoring, aggregation, analysis and modeling of system and network operations, communications, internal and external accesses, code execution functions, network and network resource conditions as well as other assessable criteria within the implemented environment. Analytical models are constructed and dynamically updated from the data sources so as to be able to rapidly identify and characterize conditions within the environment (such as behaviors, events, and functions) that are typically characteristic with that of a normal state and those that are of an abnormal or potentially suspicious state. The model is further able to implement statistical flagging functions, provide analytical interfaces to system administrators and estimate likely conditions that characterize the state of the system and the potential threat. The model may further recommend (or alternatively implement autonomously or semi-autonomously) optimal remedial repair and recovery strategies as well as the most appropriate countermeasures to isolate or neutralize the threat and its effects.

36 Claims, 1 Drawing Sheet

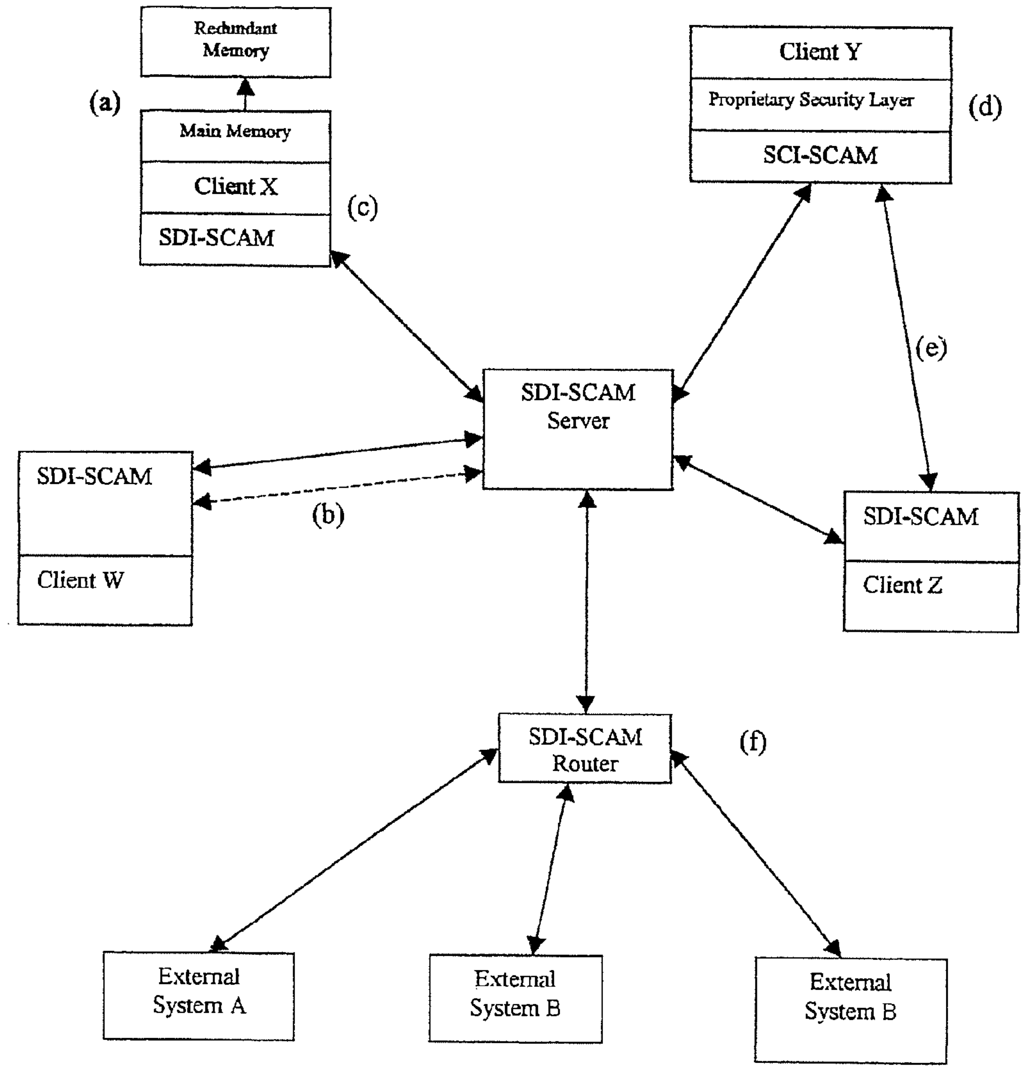


(56) **References Cited**

U.S. PATENT DOCUMENTS

6,405,318	B1	6/2002	Rowland	
6,775,657	B1	8/2004	Baker	
7,152,105	B2	12/2006	McClure et al.	
7,185,103	B1 *	2/2007	Jain	709/234
8,046,835	B2 *	10/2011	Herz	726/25
8,327,442	B2 *	12/2012	Herz et al.	726/23
8,490,197	B2 *	7/2013	Herz	726/25
2003/0002435	A1 *	1/2003	Miller	370/217
2003/0002436	A1 *	1/2003	Anderson et al.	370/218
2003/0028406	A1	2/2003	Herz et al.	
2003/0131256	A1 *	7/2003	Ackroyd	713/201
2004/0030913	A1 *	2/2004	Liang et al.	713/200
2004/0049699	A1 *	3/2004	Griffith et al.	713/201
2013/0091573	A1 *	4/2013	Herz et al.	726/23

* cited by examiner


Redundant Memory
(a)
Main Memory
Client X
SDI-SCAM
(c)
Client Y
Proprietary Security Layer
SCI-SCAM
(d)
(e)
SDI-SCAM Server
SDI-SCAM
Client W
(b)
SDI-SCAM
Client Z
SDI-SCAM Router
(f)
External System A
External System B
External System B

SDI-SCAM

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation of U.S. patent application Ser. No. 13/279,893 filed Oct. 24, 2011, which is a continuation of Ser. No. 10/693,149 filed Oct. 23, 2003, now U.S. Pat. No. 8,046,835, which claims benefit of U.S. Provisional Application No. 60/420,754 filed Oct. 23, 2002.

TECHNICAL FIELD

The invention relates to a system and method for detecting the condition of a computer network and identifying threats to the network using passive data analysis techniques.

SUMMARY

The invention is characterized by the following features:

1. Agents are installed at user (e.g. client), ISP and/or private network levels. One possible configuration enables agents to also be implemented centrally.

2. Different functions and activities are assigned to different agents. Each locally implemented version of SDI-SCAM may vary, thus not every agent/function is the same for every system implementation and not every agent is utilized for every system implementation.

3. Agents are linked together in a network. A "network" may be wide-scale distributed agent mediated information analysis and defensive response system or a small sub-network such as a coalition of organizations, a single organization and/or even a subdivision within an organization. The network may be further characterized by:

a. Redundant connections may be a prudent design consideration so that a really bad virus taking down a system will not prevent the system's agent from sending out a warning.

b. Redundant memory/hardware and associated processing capacity which can quickly become insulated from the infected and/or corrupted portions of the system may be able to function synergistically with system's analysis and the present early warning detection schemes as herein disclosed. Relatedly, such redundancy may enable a system to more readily "undo" certain processing routines and other actions if early warning detection is sufficiently expeditious and highly accurate as the presently proposed methodology attempts to achieve.

c. Architectural Flexibility is provided so as to enable communications between SDI-SCAM agents implemented on heterogeneous types of traditional computer security systems and/or with that of a centrally implemented version of SDI-SCAM.

In one very practical and efficacious incarnation of the present broad-based methodology, it is particularly prudent in many practical implementations to use the broad definition of "agent" as it is presently used within the context of the below-described system implementation to encompass (as appropriate to the particular context) security system functionality which is implemented as part of potentially any heterogeneous variation of traditional computer/network security systems that are designed to communicate with other agents that are part of SDI-SCAM and/or implement actions on their behalf. These other SDI-SCAM agents may, of course, be directly associated with that of a centrally implemented version of SDI-SCAM. Any type of associated

agents constituting SDI-SCAM may at various phases or conditions of their use possess or embody certain functions that are able to function separately, and independently of one another or the entire SDI-SCAM system or alternatively as interoperable systems within the context of SDI-SCAM's distributed multi-agent architecture. In this context, SDI-SCAM may operate and appropriately interoperate as a collection of agents that are functionally defined and whose functional purpose is exclusive to SDI-SCAM at a distributed multi-system level. Because certain informatics data collected at the machine or system level that may be of relevance to SDI-SCAM may be potentially too sensitive to release to SDI-SCAM, alternatively individual agents, e.g., representing a collection of machines, local systems (e.g., within an organization) or even organizations may for a coalition with which to exchange certain types of data based on the functional objectives of SDI-SCAM. Thus, it is conceivable that the structure of multi agent systems and disclosure rules/constraints of these agents to other agents may tend to be based upon a hierarchical structure in which data transmission and general communications from the bottom level agents to those at the higher levels tend to be more limited than top down communications, both in terms of specific data disclosure at a detailed level. Aggregative statistics and recommendations for defensive response operations, however, by top level agents tending to act more on behalf of the distributed network level and central SDI-SCAM would tend to perform communications to lower level agents, however, in top down fashion, thus making them asymmetrical interoperable systems within the larger scale context of SDI-SCAM. Furthermore, agent functionality could wherever it is appropriate and necessary provide the necessary multi system agent-agent interoperability platform, which is designed and implemented to achieve the overall functional objectives of SDI-SCAM which may itself be implemented at a local/regional level and/or a general network-wide level and in the form of either a centrally designed architectural configuration (i.e., containing some centrally located server(s)) and/or in distributed fashion, i.e., where the agent code is physically implemented at the individual system level and/or machine level. Consideration toward incorporation of such multi-system level transparency thus lends a very considerable degree of architectural flexibility to the present system.

The interoperability (or middleware) facilities that the multi-agent network provides enable the various suites of unique and enhanced anti-threat functions as herein disclosed within the present complete distributed SDI-SCAM system. However, for the sake of practical convenience of organizations and end-users which are part of the SDI-SCAM network and for the sake of efficiently utilizing potentially any/all pre-existing computer security infrastructures as installed it is an advantageous and judicious design consideration to enable SDI-SCAM architecture to possess the innate flexibility to be able to be built on top of other or otherwise existing non-distributed computer/network security platforms that may be installed in order to provide a means for synergistically and symbiotically enhancing the efficiency and rapidity of all existing detection, defensive, remedial and reparative functions of the existing system security installation as well as to add additional useful features by virtue of the system's unique multi-agent security approach. In this way, the pre-existing system security installation is able to operate independently and completely synergistically to that of SDI-SCAM while retaining its own unique and/or custom tailored functionality incorporating its own type(s) of defensive, preventive, remedial and/or reparative functions that can be passively or possibly even actively controlled or adjusted and enhanced by the present distributed multi-agent system constituting a general (network-wide) or regional (closed proprietary) implementation of SDI-SCAM. As such, a primary software-level achievement of the SDI-SCAM system is to enable the needed seamless communications between each security system and that of any other individual traditional security system's application protocol (which may, of course, include a generically implemented protocol for SDI-SCAM itself) and those protocols associated with the other types of security implementations that have been recruited for use within the distributed system architecture of the SDI-SCAM security system.

Several unique characteristics and functional features may be associated with an agent that is implemented as part of the present multi-agent distributed system. Some of the system advantages associated with such interoperability capabilities include, but are not limited to:

i. Interoperability between any heterogeneous protocols which are each associated with a security system on the SDI-SCAM distributed network;

ii. Several cardinal interoperability functions of each locally installed agent includes providing interoperability between potentially any heterogeneous system protocol which is associated with a security system and the generic system-level protocol associated with SDI-SCAM (as a protocol in and of itself);

iii. Enabling agent communications between each associated security system which is implemented locally and the SDI-SCAM distributed network (e.g., for outgoing data regarding at least certain local system-level events and actions, at a minimum); and

iv. Other implementation-level variations:

Enabling agent implementations that operate at a local system level (e.g., may be readily and practically implemented through the use of hooks or plug-ins for SDI-SCAM associated with each locally installed computer/network security system until/unless a standardized protocol evolves that supports the present distributed system paradigm; and

A non-distributed agent typically one-way (upstream) communications protocol in which an agent simply passively observes and uploads transmission data to a centralized (or other locally distributed analysis agent).

d. Inter Agent Data Exchange Based upon Data Exchange and Release Policies

The term "agent" as herein defined within the context of locally installed agents may, depending on the specific design considerations of any given local implementation of SDI-SCAM, be capable of a variety different functions including analytic functions and associated secure auditing/reporting functions (such as to human and/or autonomous operators each operating locally or as part of the broader implementation the central SDI-SCAM system). In addition, defensive and counter-offensive responsive actions to a threat may be performed in a fashion which is autonomous, manually executed or semi-autonomously executed and/or a locally implemented agent may simply perform passive monitoring and relay of auditable data from the local system. In its above-mentioned use in an analysis capacity, such as for purposes of purely informatics/data modeling and/or notification, classification, updating of the data model, etc, the actual analytic operation of such an agent may occur in a local server in distributed fashion, at a regional data warehouse (regionally distributed) or at a central SDI data warehouse. On the other hand, an agent may exist and function relatively (or completely) independently of other

agents. For example, it may only choose to receive informatics data from other agents or the general SDI-SCAM system or only release informatics data as an exclusive "data provider" to SDI-SCAM. An agent performing this function may filter only releasing certain "non-sensitive" informatics data to other agents and/or the distributed network comprising the central SDI-SCAM system. The possibility must also be considered in which the system's data collection and analytical processing of certain locally acquired data is performed by a local agent and whose external release to another agent(s) or SDI-SCAM is restricted based upon the preferences or requirements of the local system administrator (e.g., which may be revealing of secret proprietary contents, access patterns of high security clearance individuals who may be privy to certain highly sensitive files or highly secure areas of the internal network, etc.). In this case, restrictions as to release of certain data may be predicated upon certain conditions based rules such as what is the classification or characteristics of the subject data, to what entity or type thereof, is there an identified need to receive the subject data, what is the associated agent's expressed intended usage statement for that data, and does such usage require disclosure to that entity per se?

Another approach is to obscure and/or obfuscate certain key characteristics of the data itself. Certain informatics data could be exported to the constituent agents and/or the central SDI-SCAM system only as preprocessed statistics or, for example, statistical aggregates containing randomized values and/or which are ad-mixed with data from other local system sources whose identities may be concealed in order to thus obscure the actual source and/or uniquely revealing details or uniquely traceable identifying patterns or signatures of the data or its associated trail of system users.

Informatics data may also be withheld from release, e.g., to particular entities based upon any number of selected variables, a few simple examples of which might possibly include:

1. The probability of readily and practically a threat existing and/or being a certain level of severity.

2. It is determined to be highly probable that for SDI-SCAM to gain access to the subject data to be released for statistical processing with its own data statistics (i.e., using informatics data from the present local system in combination with other remote data) would result in SDI-SCAM's ability to gain statistical knowledge which could improve the present system's detection (including reduction of "false positives"), classification, defense (i.e., effectiveness in defending successfully against the present (or other likely) threats) and/or remedial actions corresponding to the possible threat. Another important consideration regarding data release is the fact that when a local agent negotiates with another agent(s) associated with SDI-SCAM for data exchange, the associated central or regional SDI-SCAM with which it is negotiating may use "strong" negotiating tactics, i.e., because it possesses statistical data aggregated from multiple sources, the potential value of this data to the local agent is much greater (particularly if the local agent's statistics are statistically correlated with the associated statistical data model of SDI-SCAM). Because SDI-SCAM's most efficient and advantageous approach involves creating an associated data exchange market, although the proposed data benefits to be provided by SDI-SCAM in exchange for the data to be released by the local agent should be weighed on a case by case basis, in the preponderance of cases the release of data by the local agent is likely to be of greatest resulting benefit (compared to the associated privacy risks) on behalf of the local agent.

Overview of Key Objectives of System Architecture

The following set of key system objectives help to address the presently presented problem which threatens system security. It is well known within the field of computer security that the importance of these objectives is matched only by the challenge that they present. Some of these include the following:

a. Detect and classify threats. Ascribe an overall probability level and send out an alert if critical probability thresholds are breached.

b. Assuming the threat exists, what are the probabilities (or probability distribution) that the threat poses certain degrees of potential danger, e.g. ranging from mild to very grave. Of possible relevance in this regard is:

i. Determine the degree of vulnerability to which the threatened system is subjected by the present threat where "vulnerability" is estimated in relation in part to the effective defensive and counter offensive means that are available and which would be effective against that specific threat which exists (or is predicted to exist). One simplistic example of a very vulnerable system is a case in which it is believed that the threat includes a hacker (or rogue script) who is predicted to possess an intimate level of knowledge (or be designed based upon the use of such knowledge respectively), i.e., as a result of intimate familiarity with the present system.

ii. Determine the predicted type of objective(s) and the overall degree of mal-intent which is observed and/or inferred on the part of the individual or organization from which the threat may be originating.

iii. Track origin of threat to likely originating entity (e.g., individual, organization, rogue script).

iv. Anticipate and if appropriate perform appropriate notifications as to which other individuals, organizations and other nodes are likely to have been or are currently being subjected to the threat.

Examples of classifier attributes that may appropriately exemplify some of the types of threat classifications criteria suggested above (which are typically of a probabilistic type) may include, but are not limited to:

Likely/potential individuals or coordinated groups of individuals;

Likely/potential organizations or interests represented by the threat; and

Relatedly, the possible underlying intentions or objectives of this associated underlying entity or interest.

c. Alert and Notification Features—The warning system provided by SDI-SCAM may be activated in response to any variety of possible criteria of notification indicative that the security of the present system may have been compromised (which may, in turn, be variables affecting the overall probabilistic determination for both exposure to and actual infiltration by a given threat). These may include (but are not limited to) the following parameters (as weighted attributes):

i. Probability that the communication with an entity has occurred (given the probabilities);

ii. Probability that the system/entity which has communicated with the present system/entity possesses a given threat(s) (given the probability of i).

iii. Probability of a known or unknown communication(s) imposing a threat upon the present entity (given the probability that the communicating associated known or unknown entity possesses the threat as suggested in ii).

iv. Probability that a given threat to which the present system/entity has been exposed has actually violated and/or compromised the integrity of the present computer system or network (given the probability of iii and given the defensive

characteristics of the present system relative to the particular type(s) of possible threat(s) which are presently likely to exist).

d. Select and deploy an optimally suited defense scheme customized for the threat based upon all determinable behaviors, characteristics, and conditions of the threatened system(s) as well as those of the identified threat, provide an appropriate remedial countermeasure based upon the same criteria.

e. Document all observations regarding the type of structure and functional characteristics of the intrusive entity and that of the system(s) with which it interacts, update/refine statistical models accordingly, in order to optimize overall system level intelligence so as to improve the above described operations for detection, classification, tracking, determination of origin and intent as well as associated defensive/counteroffensive measures, etc. The above system objectives are most effectively achieved by implementing a version of the system wherein sharing of critical data can be performed in distributed fashion and can be performed by and between the most likely relevant servers and networks at the most critically important times and in dynamic fashion. Furthermore, the above system objectives are most efficiently achieved by employing statistically-based predictive modeling techniques which are themselves implemented in the form of an associated distributed architecture which is capable of dynamically responsive network-wide remote polling, statistical processing and data redistribution/updating capabilities.

BRIEF DESCRIPTION OF THE DRAWINGS

The FIGURE demonstrates some of the architectural features of the invention, including (a) redundant memory within a given machine, (b) redundant connections between clients and servers, (c) SDI-SCAM installed as a primary security system, (d) SDI-SCAM piggybacking on an existing security system, (e) direct client-to-client agent communications, and (f) on a router.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

The algorithms implemented by the invention include:

a. Detection-Bayesian Belief Network

b. Prediction of behavior (e.g., of a virus or worm)-Bayesian Belief Network, Statistical techniques for modeling patterns based on sequentially occurring behavior patterns.

c. Classification:

i. Bayesian Belief Network

ii. Pattern matching nearest neighbor (in order to determine closest category based upon similarity of its constituents).

iii. Statistical techniques for detecting patterns based on sequentially occurring events.

d. Immune Response of System

The features and attributes of the resulting system include:

i. Actions of virus.

ii. System (or SDI-SCAM) responses to actions of virus.

iii. Actions of a hacker.

iv. System (or SDI-SCAM) responses to actions of a hacker.

v. Discrete objects as targets of the threat:

people

network/organization

server
file
content
existing worm or virus
classes for each of the above
data release policy for each of the above
vi. Textual and other multi-media content features
vii. Classification of the threat
viii. Potential triggers of the virus or threat (e.g., activation, replication, communications, exportation, importance, etc.)

Links Connect to Servers (e.g., Data Warehouse(s), Gateways and/or Peers)

The SDI-SCAM servers perform the following functions:

a. Collection and analysis of distributed data—Servers filter and analyze incoming signals, collecting pooled information from agents.

b. Notification to multiple agents—If warning or alert is warranted, server sends it back down the line to agents. In one variation, the present multi agent system functions effectively as a truly distributed agent architecture such that probabilistic threat suspicion model continually filters, analyzes incoming signals while dynamically redistributing its analyzed data and probabilistic updates reciprocally back to its associated constituent agents.

Example Case Implementation of the Preferred Embodiment From a Procedural Perspective

1. Agents report any suspicious activity/data that exceed suspicion threshold.

2. After any scam/break-in/viral infection is confirmed agent transmits as much log/content/behavioral tracking information as possible in order to determine origin of the problem.

3. Data related to suspicions and confirmed attacks constantly sent back to servers.

4. Servers scan for patterns, correlate with news wires, sysadmin communications, anti-viral and other external databases.

5. Servers determine definite/probable flags for various problems and formulate solutions (human in the loop would be very useful here—sysadmins who have recovered their systems could submit defensive fixes and strategies).

6. New detection routines/security patches/warnings/defensive tactics uploaded to agents. May want to implement some level of security so that potentially compromised systems will not be given access to pooled information (as this might give a hacker information on what is known about him).

Exemplary Applications of a Few Primary Types of Threats Which Must be Recognized and Counteracted

1. Hacker break-ins (script kiddies/corporate/terrorist)—Intrusion Detector such as ASID. Pattern analysis may be performed. For example, are similar break-ins/attempted break-ins/peculiar noise being spotted across network? If so, such observations may signal novel or concerted approach to breach security. Thus, distributed and dynamic agent communications among widely distributed agents is particularly advantageous for the above reasons. However, by contrast, newly appearing completely novel threats may appear and the associated possibility of such considered and the associated possibility of such considered, e.g., first time release of a new virus hacker attempt or cleverly crafted stealth break-in attempts or rogue software infiltrating the system by "corporate insiders", (e.g., manually installed rogue software scripts, etc.).

2. Viral infiltration may be recognized and counteracted by:

code analysis
behavior analysis
data, language, content analysis
Protections against mutation/dead ringers
Pattern analysis
Other Wolfram-inspired ideas

Because rogue scripts may infect a system and present themselves in a dormant mode (before or after activation mode), the above suggested system objective of documentation and associated reduction thereof to a statistical model of previous intrusions and infections can be used to better anticipate the "triggers" from active to passive modes and vice versa as well as the condition/combinations thereof which most likely had occurred within the context of active threats, assaults, intrusions, spreading, etc. to the system as well as reconstructing the conditions/combinations thereof that are likely to have accompanied any/all of the above events (i.e., modeling and predicting the epidemiological characteristic of the rogue script). Because of the unpredictable and potentially malicious behavioral characteristics of rogue software, it is of additional protective benefit to the system to further insulate the agent layer of the system so that control of it is not seized by the potential threat.

In developing a practical statistical model used in virus defense and remedial actions it is useful to consider various attributes like what attributes of a vaccine were effective against previous viruses possessing the same/similar attributes (of course, these features may be further updated and modified once installed as additional feedback is received regarding its effectiveness and associated behavioral responses, for the vaccine and its associated countermeasures).

3a. Untruthful commercial messages (e.g. Nigerian scam): that is, signal itself is not dangerous, but content is.

3b. False information/rumors/Hoaxes
Statistics/NLP
Warnings from member agents

Cross-check content against recent news stories discussing scams.

Cross-check names and email addresses in messages against known criminals/hoaxsters and aliases for both/either.

Semantic Analysis (for language patterns consistent with trickery/appeal to risk taking tendencies, naive tendencies and other weaknesses that would predispose individuals to predatory individuals and their associated scams as well as messages directed towards such individuals based upon predisposing characteristics which would potentially be accessible to such individuals (e.g., the elderly, the uneducated), those of limited literacy skills (i.e., to the native language or to effective us of a computer), the persons of status, the unemployed or under-employed, teenagers and those who have been receptive to similar unscrupulous solicitations or hoaxes in the past or otherwise exhibit predisposing characteristics which can be predictively correlated to vulnerability to certain particular types of scams which present statistical relevance to SDI-SCAM.

Other behavioral Data

Overview of Hardware Architectural Configuration

The architecture component of the present technical methodology as illustrated in the FIGURE should be viewed with appropriate consideration for optimal flexibility and inclusiveness of any viable configuration of hardware and software components as is needed to perform the presently described functions in distributed dynamic and scalable fashion. However, due to the unique nature of the application-level context of the present system (which likely attributes a substantial need for greater localized control and security) to any/all data, which is proprietary to the local system or network, the use of a fully distributed server architecture and/or local distributed central data warehouses (defined within SDI) may be particularly appropriately implemented as preferred network architectural variation.

Distributed Architecture Details

Ideally, as indicated, all of the probabilistic (such as Bayesian) analysis and modeling techniques are performed in fully distributed and dynamic fashion thereby assuring that the system's distributed data modeling and associated early warning detection facilities are optimally prepared and familiar with current threats, scams and their most likely associated objectives and modus operandi. In the preferred embodiment of the distributed architecture, all of the nodes on the system periodically become updated vis-à-vis the enabled features of SDI-SCAM's distributed architecture, collecting, pooling and redistributing statistics to and from other agents. If/when a local server's suspicion threshold becomes elevated the updating process is again triggered and thus may typically be automatically carried out among those agents. The present updating process may thus be elicited by triggering of the suspicion threshold at one of the constituent nodes or standard periodic updating. In the former case, a preferential priority should be prescribed by the system in accordance with an overall suspicion level based upon certain criteria which are predictive of any given constituent node on SDI-SCAM's multi-agent distributed system having been subjected to the threat. These preferential priorities may in turn be variables used in an overall formula for:

a. Determining whether and to what degree the need exists to poll a given agent.

b. If so, determining the most appropriate sequence/prioritization of polling and associated selection of server(s) and type of analysis technique which appears to be mandated based on feedback from the currently polled agent(s).

A hierarchical scheme addressing this latter issue is disclosed the methodology for performing statistical data collection, and updating in the most efficient manner possible within a distributed architecture as described below. Based upon the system's determination of predicted level of suspicion for a threat, remedial measures in turn may be accordingly indicated, (e.g., a custom dynamically created vaccine could be developed using techniques based upon adaptive rules which utilize inputs from the distributed agent which possess potentially all forms of relevant data).

As listed below, a few example criteria are provided which may play a role in affecting determination of the need to poll a given remote agent's corresponding node based upon a presently suspected threat. Polling priority may include (but is not limited to) those nodes which the presently threatened node is likely to have recently communicated with directly or (often) indirectly (based upon past communication statistics and network flow patterns) in order to determine whether they have been exposed to a particular threat which may exist on a given client network node. A few pre-disposing factors may include:

i. Those which have recently communicated between each other or among one another;

ii. Those which have recently established a communication link with other nodes which are either the most identically shared or otherwise "related" to those with which it has established communications so as to maximize the system's ability to predict those nodes for which there has been the most common communication connections with other (identical) nodes. Similarly, considering in this model the history of which nodes and connections therewith have eventuated in associated threats (e.g., as directly or secondarily conveyed).

iii. The greatest degree of "similarity" or alliance to the present entity (e.g., corporate or strategic affiliation/alliance, category similarity, commercial/economic or technical interdependency underlying the business relationship, etc.). Other nodes then secondarily may be updated and reciprocally polled for up to the minute statistical updates as well. These example criteria which may be indicative of potential "similarity" with another node(s) that is suspicious may be of predictive probabilistic value in presaging a certain degree of similar suspicion in the other nodes because:

a. Such common characteristics may be suggestive that recent communications were received from a common source or were exchanged between each other;

b. The associated threat was intended to target systems or servers possessing certain defined characteristics.

The structure of the preferential priority scheme for updating nodes across the entire distributed network which constitute the presently proposed tree-like acyclic graph configuration of the updating scheme (which uses a preferential prioritization scheme using a decision tree) could be constructed with consideration to a variety of exemplary criteria. The updating process within the present context may be elicited by the need to poll a node for further information based on the probability of subjection to a potential threat where key data used to determine this probability may include other nodes with which the present node did or may have communicated (probabilistically) and/or notification data received directly from the node's associated agent. Upon polling and statistical analysis of certain key data from the subject node of possible concern by SDI-SCAM typically conclusions derived from the analysis are transmitted back to the subject node as well as possible recommended defensive and/or remedial responses to be executed by the local agents and any/all appropriate statistical updates which need to be performed at the local level as well as at the general level for SDI-SCAM. Alternatively, it might be the case that some or all of the data on the subject node to be otherwise slated for polling in accordance with the present associated conditions is restricted from disclosure to SDI-SCAM. In such a case the restricted data is instead analyzed by the local data analysis agent. It may be combined at the local analysis agent or (subject to the local agent's data disclosure policy regarding the associated processed data) and it may be uploaded to the analysis agent associated with SDI-SCAM for this purpose. It may, however, be the case in certain instances that SDI-SCAM is only able to recommend certain defensive and/or remedial responses if the data disclosure policies of individual local agents contributing to SDI-SCAM's statistical model do not permit the redistribution of statistical data currently possessed by SDI-SCAM which was derived from their proprietary local sources.

The method for developing an adaptably self-configurable optimized acyclic graph for statistical data updating/polling based upon a hierarchical structure is disclosed in issued U.S. Pat. No. 5,754,939, entitled "System for Generation of User Profiles for a System for Customized Electronic Identification of Desirable Objects", and is used in the application context of polling for features and their associated (updated) scalar values as associated with user and target object profiles. The description of that patent application is hereby incorporated by reference. The use of a hierarchical cluster (or alternatively a decision tree is described in this specification) to poll potentially physically distributed nodes for data may be required to add statistical confidence to a portion or branch of the tree (represented as an acyclic graph for which there presently exists the condition of sparse data). By direct analogy, this technique could be similarly utilized in the case of the present system, however, in which case the degree of statistical uncertainty (previously measured by the measured degree of sparse data) in the present adaptation be represented by the degree of suspicion as measured at any given node across the distributed agent architecture. In this regard it may be necessary for SDI-SCAM to exchange further detailed level variables with that of the local agent in order to achieve a satisfactory level of statistical confidence regarding whether a potential threat is attempting (or has attempted) to intrude the present system, in addition such a hierarchical decision tree may also introduce to the threat certain selected stimuli and/or emulation of system characteristics so as to be able to elicit associated behavioral response actions on the part of the threat so as to ultimately ascertain an optimal level of statistical data regarding the most important variables in use by the system to classify and quantify the nature and severity of the potential threat (discussed within the present disclosure) which are accordingly incorporated within the queries which constitute the present decision tree.

In this regard, for efficiency's sake, in one preferred variation it is desirable to utilize a particular scheme to quickly poll the feature vectors and other attributes used in the statistical data model based upon a preferential prioritization of those features which are the most relevant to the probabilistic determination of the probability of infection (or subjection to another threat) as well as its likely degree of rogueness. One could envision the efficient use of a decision tree as part of the distributed system which may, for example, be designed to poll agents in order to most rapidly determine the presence and/or discover data relating to the following parameters with regards to providing further useful data to the present predictive data model. Some of those criteria for use in determining preferential priority in SDI-SCAM establishing communications with and updating nodes across the network include:

i. The factors discussed in items i-iii above which include among others:

Patterns of communications particularly very recently with other nodes, which are determined to be the same or deemed "similar".

Communications (particularly very recently) which have been established between those particular agents.

ii. Those nodes which are part of or associated with in some way the same internal network, e.g., sharing physical hardware components, geographic proximity, common individuals who directly or vis-à-vis an associated organization affiliation possess common access to the present server or system of subject concern.

(iii). Networks that are particularly vulnerable or have particular holes to be potentially exploited by would-be intruders.

(iv). Networks that contain particularly important and/or sensitive data and/or the consistency and integrity of whose operations are particularly critical for one reason or another, (e.g. economic, financial, utility infrastructure, national security, etc.

(v). What is the likely source and intended destination(s) or types thereof (if any) of a particular suspicious script(s) or hacker(s)?

(vi). Are there any common characteristics or common patterns, transmission/distribution patterns, etc. which exist between different suspicious entities (either the above temporally or otherwise).

(vii). Based upon existing transmission patterns/characteristics, what are the likely destinations that a particular suspect entity likely is, has been or will go (thus forming the basis for destinations which may likely be intruded).

(viii) Attributes (if any are present) which if determined to exist or exceed a certain threshold value effectively eliminate the associated system as a likely candidate for intrusion or viral infections (thus saving time/bandwidth costs associated with modeling and tracking the suspect intrusive entity).

(ix). Attributes which will most quickly/effectively achieve a determination of whether it is "sufficiently likely" that the system has been intruded. At which point, if it is deemed judicious to do so, the decision tree may probe further at a deeper, more comprehensive level of data acquisition in order to attempt to detect the potential presence of commonalities with other likely systems that are likely to have been intruded by the same entity (much like the director of an investigation of a potential criminal event that may be linked to the suspect of a set of serial crimes). Likewise, for a suspected intruder which has been analyzed and profiled to the greatest extent possible poll from those other agents at a high level of detail, which are likely to possess highest degree of historical data pertaining to that particular suspect rogue entity (or type thereof), e.g., as indicated by associated statistical data regarding the rogue entity and associated defenses and remedial techniques that were successful or unsuccessful.

(x). What are the computer system/network (or associated security) characteristics and/or actions or responses (and weighted statistical values which may have elicited the present (or present type of) virus (or other threat) to perform certain actions relating to other systems that were harmful to the present system (or other systems, e.g., as a result of eliciting further spread or mutation of a given virus).

(xi). Are there properties of any given piece of code which have been seen to propagate across the network and/or multiple locally or remote fashion or demonstrate other characteristics of rogue or unwelcome artificial life?

(xii). What piece(s) of code believed to be normal appear to demonstrate the highest probabilistic degree of similarity to a virus (thus code sequence may then become a template to perform deeper analysis of those other pieces of code on local or remote systems to which this heightened probability can be attributed)?

(xiii). Does there exist and if so, what is the sequel of code which contributes (statistically) to the determination of elevated level of suspicion of a threat on another system within SDI-SCAM?

c. Routine maintenance and upgrades to agents done by server. One specific architectural variation in which the associated architecture is devoid of any centralized agent functionality, i.e., it is entirely distributed, will now be described.

General architecture—agents are nodes, servers are hubs.

Scam Detector—Much has been written recently about a variety of annoying and even harmful information disseminated throughout the Internet that ranges from simple propagated rumors, misinformation and inaccuracies to deliberate hoaxes or fraudulent scams with malicious intent to profit at the expense of other people who are duped into believing deceptive promises, claims and other information. Some of the most insidious of the latter include the notorious Nigerian bank account scam, aid to US soldiers in Afghanistan, aid to victims' families of the 9/11 tragedy and a variety of charity based scams. Unquestionably, the most abhorring, and in fact, disturbing form of scam involves those dangerous individuals who exploit use of the Internet's very privacy protecting advantages in order to pose as a type of individual (e.g., a teenage girl) which they are in fact not (e.g., while in fact being a 40 year old stalker or even sexual predator of children). In order to address these problems both individually and collectively, what may be needed is a system which may be implemented at the browser or ISP level, which collaboratively and innocuously combs through both specific content and users' behavioral responses and information oriented responses to such information. Accordingly the system is based upon a statistical model containing statistical and NLP components and operates in a fully distributed and collaborative fashion. It observes and compares information using statistical NLP in order to determine the suspicion thresholds of any given content which fits the basic format of a potential scam. The language model may be based upon a set of adaptive rules which are initially manually inputted and which become refined and modified in accordance with relevance feedback. Examples of sources of these rules may include statistical models of "deceptive information" (perhaps from a training corpus). It may also be based upon other pre-existing scams, which have been clearly identified as such. Of course, there are many subcategories of scams which fit the definition of a scam and each would be modeled individually, for example, false or exaggerated claims made by spam advertisers (i.e., false advertising) traditional Internet scams, Internet rumors or other false information which could become propagated, etc. Although it is not an extremely likely scenario, such a system could also be used in a protective capacity in which, for example, some rogue entity were somehow able to gain control over the network (e.g., cyberterrorists) and disseminate apparently legitimate information that could result in a panic or frenzy and/or such entity posing, for example as a government authority figure requesting that individual (or the public) to react in a way that could be particularly harmful to an, entity, government, (e.g., an individual, such as a person/leader in a position of authority, a group of people, or an entire nation's national homeland security interest), or for example, such a similar type of system wide seizure could also, for example, be used as a medium through which individuals could be duped into inappropriate disclosure of highest confidential or classified information to the wrong entities or at a system level, convincing another system that appropriate actions permissions which the seized system has access to is sufficient evidence that requested sensitive information is being released to only appropriate individuals, besides the extremely unfortunate and contemptible efforts of unscrupulous individuals to prey upon the fears, concerns, and sympathies of the unsuspecting public in times of tragedy and/or associated fear. What is potentially equally as worrisome is the possibility of such individuals to do further damage, for example, by:

1. Posing as a legitimate (though in reality a surreptitious) organization or individual and causing panic by either initiating a believable rumor or compounding fears which individuals may already have immediately after or during a tragic event or before such a feared tragic event even occurs. (One could easily imagine this very scenario, having taken place immediately following the 9/11 tragedy had the Al Qaeda network had additional funds and/or IT resources at the time). Such hoaxes could be any or all of the following: currently prevailing political concerns (such as threats by feared terrorist networks), particularly surrounding potentially explosive or unstable situations; rumors already circulating or benign versions of existing communications (which are considered "legitimate") transmitted via the news media, the Internet, etc. during or right after a tragic event; and/or taking over a data communication network (which may include electronic news media which is recognized as legitimate and credible).

In addition to content based input, the system may usefully further leverage user response behavioral data as well as content provided by the user in response to the information. The system may also attempt to validate or authenticate the information by tracing it back to its original source, e.g., was it propagated by multiple sequential users, did it accordingly evolve much like the characteristics of a rumor or did it originate from a single individual? If the latter is true, what is the trustworthiness (ultimately, believability) of the individual or organization from which the information originated (e.g., which could be performed by conducting an instantaneous background check on both)? Also, the co-pending U.S. patent application Ser. No. 10/202,302, entitled "Database System for Identifying Potentially Litigious Patients" with a few obvious modifications could be customized for predicting instead of probability of litigious activity, rather probability of an individual to engage in questionable business practices or actual fraud (even if an explicit previous history of the same is not apparently evident at present). This may include the other (legitimate) activities the user is involved in. Corroboration-based techniques may also be useful, e.g., to what degree does the information corroborate with other relevant information sources whose trustworthiness is quantified or what is the degree of corroboration of these other information sources with the present one particularly with relation to similar disclosed information in the past. What is the degree (if any exists) of user complaints about a particular piece of content or information source ("degree" includes prevalence and severity of alleged inaccuracy, untruthfulness or impropriety)? Certainly an appropriate rational approach to determining the nature of a likelihood of content to be of a damaging nature would be to determine these probabilities based upon the characteristics of the content or script itself as well as the identity or likely (inferred) identity(s) of possible individuals and/or entities with whom they are associated. It is, of course, important in this probability determination of rogueness of the source, to not only predict the probability of the entity and/or individual being the originating source of the data, but also what is the associated probability of such individual or entity possessing certain characteristics which themselves comprise an estimated degree of rogueness which can be predicted for its associated data.

In an even more advanced variation of the present scheme, it may even be possible to develop a rule-based scheme for probabilistic analysis of a piece of content based upon relatively comprehensive analyses and custom rules which assess the typical and semantic language style characteristics and (importantly) deep conceptual meanings which could be indicative of potential scams (which, for example, take into consideration, factors such as degree of financial opportunity presented, ease of attaining the associated financial rewards, areas in which individual might be particularly interested improperly or erroneously informed or gullible as well as other motivations, intentions or psychological factors common to the domain of public persuasion), which might motivate a scam artist to contrive such a scam based upon his/her thought processes as well as potential expected rewards and the structure of previously successful scams.

Few highlights and key thoughts.

1. The national security application incorporates many of the same functional features of the scan detector.

2. At a high (diagrammatic) level a system security adaptive learning based approach in accordance with the invention utilizes a rather broad range of inputs which include, for example:

a. Code analysis;

b. Behavior analysis, such as sequence of system behaviors that may be internal or external to the system or of external thus attempting to counteract its own counter measure for detection (i.e., vis-à-vis mutation).

b. The "dead ringer's" scenario (code emulation code)—another important countermeasure attempted by the clever virus is attempt to assimilate all of the fundamental structural (code sequences) and functional characteristics of the surrounding code in which it is embedded in order to make:

i. Its code presence more invisible and thus less detectable;

ii. Its functional behavior less detectable (from that of the authentic code's functionality).

c. The "dead ringer's" scenario (code emulating humans)—The clever virus may further attempt to create a simulation model of the functional aspects of semantic language and even a simulation model of specific human beings and moreover, those of actively communicating humans (e.g., citing the case of an email intended to congratulate one's boss but instead it may be laced with insults). Likewise, if semantic models of specific humans can be created to the extent of passing the Turing test and thus disrupting human personal and business relationships and processes then almost all other types of human behaviors could be equally readily simulated.

Practical Considerations—The virus's ability to acquire complete knowledge about the surrounding code should not be underestimated.

a. One practical consideration is that such viruses can readily observe explicit behavior of the surrounding code in which it is embedded and it would appear to be the case that it could also diffusely embed itself in such a way that its ability to scan the structure of surrounding code sequence could also be performed almost certainly in a completely stealth fashion (thus emphasizing the need for analysis of behavior content and code).

b. Embed within the standard code itself some variation of the presently described scheme in metadata-type format which could perhaps function in very similar fashion to that which is presently herein, and perhaps use a decision tree to query the code sequence and its behavior as it begins to execute thus enabling the present system to detect problems, perhaps at an earlier state. Thus, instead of taking periodic "snapshots" of the code, numerous "snapshots" would be taken of it on a continual basis.

c. Steven Wolfram states that for many types of code it is possible to query and determine certain fundamental (high level) characteristics for even code which exhibits fairly complex forms of behavior. This would imply that the functional objectives or the present system's intrusion detection function could be ideally/optimally achieved by a hierarchical querying scheme such as a decision tree.

Because analysis of code, behavior and content is occurring, a statistical methodology would be very well suited in terms of its ability to capture and leverage a wide variety of types of feature-based statistical inputs. The invention is further characterized by techniques that are capable of detecting non-linear as well as standard linear relationships (using, for example, non-linear kernel regression method). Specifically, patterns of complex code sequences are linear; however, the characteristics of the actions, elicited directly therefrom, correlate with them in non-linear fashion. Likewise, sequences of actions (more complex behavior) will likely correlate non-linearly as will content features of text or spoken language.

Other Design Considerations, Practical Problems/Issues and Their Associated Technical Approaches, Applications, Etc

a. Applying Techniques That are Cited by Stephen Wolfram—

One of the interesting and practically relevant experimental observations made by Stephen Wolfram in his book, "A New Kind of Science" is the fact that for the vast preponderance of software code (generally speaking) which exhibits demonstrably, complex behavior, the underlying rule set which governs that particular code is remarkably simply in nature. This presents an opportunity by which this simplicity can be exploited to the advantage of an observer who can pose certain queries about the fundamental nature and characteristics of the code and how it is likely to behave under various conditions. Although these queries and their associated answers may be (and usually are) limited to a general and non-specific level, it is conceivable that one could develop an algorithm which is designed to query and retrieve information about other code (which is vulnerable to infection by rogue viruses) or even, to some much more limited extent, to the "software" underlying the behavior of biological agents (i.e., hackers) to the extent that their simulation model can reduce their behavior and underlying motivational objectives to that of a computerized simulation model. The functional design objectives of such a system would likely be divided into two components:

i. A purely observational (passive) feedback based component; and

ii. A response based approach by which certain behaviors are elicited based upon certain system-generated queries which elicit the behavioral response. It is reasonable in this approach to apply a decision tree which preferentially selects dynamically and reconfigurably selective environmental system conditions and/or stimuli to which the code is subjected in order to rapidly acquire the most relevant and informative information possible based upon the present collection of knowledge known about the entity at the time of each respective stimulus.

Trusted Server

Different servers or databases can benefit from sharing and exchanging information with one another about attacks. Suppose that a server is attacked by a virus. That server can benefit other servers by publishing how that virus penetrated into the system (a particular email message or a web site for example). Yet if the server were to publish this information, the server gives away the information that he was attacked by a virus. Such information can be damaging to the server. Therefore, the Server would like to keep such information privately. However each server would benefit from warnings of what the viruses look like before the server is attacked. This way the server can avoid reading certain emails or accessing certain web sites.

One way for the servers to share information about attacks securely without revealing information about themselves is to use a Trusted Server. The trusted server will be a paid server different from the others. The only purpose of the Trusted Server is to help the other servers communicate with one another without revealing information about themselves. The Trusted Server is chosen in such a way that it is trusted by all other servers. In other words, the Trusted server should have no motivation to harm the privacy of any of the other servers.

The protocol for the Trusted Server would be the following: (1) Each server sends to the Trusted Server the attack information available to it, (2) The Trusted Server then gathers this information and sends the warnings to the servers. This way none of the servers except for the trusted server know where the information is coming from and which server was attacked.

The trusted server has to ensure that it does not distribute false information to the other servers. This can happen when an adversary can pretend to be a server that was attacked by a virus and pass misleading information onto the other servers. This could cause some servers to avoid reading important emails that are not harmful at all.

There are several ways to avoid this. First, the Trusted server (TS) can request to see the emails of the servers. This way the TS can run them on some machine and see the results. The TS can choose to run each with some probability. This way the chance of finding the adversary are large, yet the work that the TS has to do is minimized Second, the TS can decide to announce only those warning that arrive from several servers above some threshold. This threshold can be determined based on the application. If the servers are willing to get more warning, then the threshold will be low. If, on the other hand, they want to make sure that the warnings they are getting are real threats, then the threshold can be high. For this to work, the servers can report anything that looks different than usual on their machine. That is, if they receive an email from an unknown party they can report it to the trusted server before opening it. If the TS notices that many servers got that email, then the TS sends a warning. If a server does not get a warning, then they can assume that it is not likely to be a wide spread virus. Third, the TS can only use the "reputation" of the servers in deciding whether to broadcast their warnings or not. Each server can come into the network with a certain good reputation that is recommended by another network. Another alternative is for a server to build its reputation from scratch depending on the behavior which is judged by the TS. Thus, servers which only send helpful warnings will gain good reputation and more of their warnings will be broadcasted. On the other hand, an adversary can lose its reputation once and be prevented from disrupting the network again.

The Trusted server can also create honey pots in order to track the adversary. In this case a honey pot would be a trap that would cause the adversary to disclose himself because he claims some warning about a virus was true.

Redundancy

Redundant memory/hardware and associated processing capacity can be kept insulated from the infected and/or corrupted portions of the system. This can be done by keeping two copies of the system one would be "active" and online. The other would be stored as a backup which is not used. However the backup will be an exact copy of the original so that it can replace the original in a modular way. The replacing can be done during off peak hours. When the "active" system is replaced by the redundant one they need to be compared for their differences. This has to be done for two reasons. First the redundant system has to be updated to become up to date as the "active" system was. Presumably there are many changes that occurred through the active time. Second, this difference check can also aid in discovering attacks. This is especially so in systems that do not have many changes during one time or that have changes that are predictable and always of the same type. Therefore, when the active system is exchanged with the redundant one, only the non-virus changes will be transferred.

Trusted Server and Statistical Analysis

There are a few ways to detect which are the harmful virus changes that are made to a system. One is via an interaction with a Trusted Server as was described above, another is by keeping a statistical record on the behavior of the system, the kind of changes usually made, important components of the analysis would include amount and type of changes, a list of changes that accompany a particular change, and a list of changes that were found to be harmful in the past.

In addition, the information obtain from the Trusted Server will also help in refining the statistical analysis. The trusted server can give a clue as to which patterns to look for in the data which would help reduce the search size considerably.

What is claimed:

1. A distributed network security system that detects the state of a computer network having a plurality of nodes including identifying potential threats to the computer network, said system comprising:

at least two agents disposed in said computer network that collect data representative of operations of said computer network including respective nodes in said computer network, said data relating to communication, internal and external accesses, code execution functions, code analysis and/or network resource conditions of respective nodes in said computer network; and

a server programmed to:

compare data collected by said at least two agents to determine code analysis and/or activity models characterizing conditions within said computer network including behaviors, events and/or function of respective nodes of said computer network, said behaviors representative of normal states and one or more abnormal states representative of suspicious activity indicative of an attack or threat to said computer network,

perform a pattern analysis in the collected data to identify patterns in the collected data representative of suspicious activities indicative of an attack or threat to said computer network and to develop code analysis and/or activity models from the collected data representative of activities of said computer networks in a normal state and activities of said computer networks in an abnormal state based on said identified patterns, wherein said pattern analysis involves comparing data collected by each said agent to the data collected by another agent to identify similar patterns of suspicious activities indicative of an attack or threat to different portions of the computer network, and

determine during said pattern analysis if a probability threshold for detecting and classifying a threat is breached by said similar patterns of suspicious activities and, if so, send out an alert to other agents, a central server and/or human operator.

2. A system as in claim **1**, wherein said server protects individual machines in said computer network by pooling and analyzing information gathered from different machines across the computer network by said at least two agents passively collecting, monitoring and aggregating data representative of activities of said plurality of machines.

3. A system as in claim **1**, wherein said alert includes a probability or probability distribution that a threat poses certain degrees of potential danger, threat classification or nature of said threat from analyzed data in order to determine an origin of suspicious activity indicative of an attack or threat to the computer network and potential counter measures from other individual machines in said computer network.

4. A system as in claim **3**, wherein said alert includes a probability of likely threat subjection to other identified individuals, organizations, nodes, degrees of potential danger, threat classification, and/or nature of said threat.

5. A system as in claim **3**, wherein said alert is accompanied by an optimally suited defense scheme customized for the threat based determinable behaviors, characteristics and/or conditions of the threatened system in order to provide appropriate remedial counter measures.

6. A system as in claim **5**, wherein said alert includes defensive and counter offensive responsive action protocols appropriate to said detected likely threat.

7. A system as in claim **3**, wherein the analyzed data is provided to other agents for purposes of updating their data models.

8. A system as in claim **6**, wherein said defensive and counter offensive response to said likely threat are performed in a fashion which is autonomous, manually executed, or semi-autonomously executed whereby an agent performs passive monitoring and relays auditable data from a portion of the computer network local to said agent.

9. A system as in claim **3**, wherein said alert includes information that is recommended for remedial repair and/or recovery strategies to isolate or neutralize the identified potential threats to the computer system once the existence of harm to the system has been confirmed.

10. A system that detects the state of a computer network having a plurality of nodes, said system comprising a plurality of distributed agents designed for adaptive learning and probabilistic analysis, said agents passively collecting, monitoring, aggregating and pattern analyzing data collected by respective distributed agents to identify similar patterns of suspicious activities indicative of an attack or threat to different portions of the computer network, determining from said pattern analysis whether a probability threshold of suspicious activity indicative of an attack or threat to said computer network has been exceeded, and when said probability threshold has been exceeded by said similar patterns of suspicious activities, alerting other agents, a central server, and/or a human operator.

11. A system as in claim **10**, wherein said adaptive learning is implemented by a Bayesian analysis system.

12. A system as in claim **10**, wherein the said adaptive learning is implemented by a pattern matching algorithm.

13. A system as in claim **10**, wherein said distributed agents are part of a distributed system architecture.

14. A network threat and response system comprising a plurality of distributed agents that collect, monitor, aggregate and pattern analyze data representative of behavioral activities and/or code in various locations across a computer network collected by respective distributed agents to identify similar patterns of suspicious activities indicative of an attack or threat to different portions of the computer network, and that determine from said pattern analysis whether a probability threshold of suspicious activity indicative of an attack or threat to said computer network has been exceeded by said similar patterns of suspicious activities, and when said probability threshold has been exceeded, alert other agents, a central server, and/or a human operator, wherein said distributed agents together form a scalable distributed network architecture capable of dynamically responsive

network wide remote communication, statistical processing, data distribution and redistribution and updating of said distributed agents.

15. A system as in claim **14**, wherein said scalable distributed network architecture is adapted to provide rapid communication, detection, classification, tracking, probabilistic analysis and/or defensive and counter offensive measures.

16. A network threat and response system comprising a plurality of distributed agents that collect, monitor, aggregate and pattern analyze data representative of behavioral activities and/or code in various locations across a computer network collected by respective distributed agents to identify similar patterns of suspicious activities indicative of an attack or threat to different portions of the computer network, and that determine from said pattern analysis whether a probability threshold of suspicious activity indicative of an attack or threat to said computer network has been exceeded by said similar patterns of suspicious activities; each of said agents performing adaptive learning in order to detect the probability, classification and/or nature of a threat to said computer network, to develop notification thresholds for alerting other agents of potential threats to said computer network, and to develop appropriate counter measures, including defensive, remedial and/or reparative functions as well as preventative functions, wherein at least one of said distributed agents further performs at least one of alerting, auditing and/or reporting functions to human and/or autonomous operators.

17. A system as in claim **16**, wherein each of said agents provides defensive functions whereby redundant memory, hardware and associated processing capacity are insulated from an infected or corrupted portion of the computer network.

18. A network threat and response system comprising a plurality of distributed agents that collect, monitor, aggregate, and pattern analyze data representative of behavioral activities and/or code in various locations across a computer network collected by respective distributed agents to identify similar patterns of suspicious activities indicative of an attack or threat to different portions of the computer network, and that determine from said pattern analysis whether a probability threshold of suspicious activity indicative of an attack or threat to said computer network has been exceeded by said similar patterns of suspicious activities, and when said probability threshold has been exceeded, alert other agents, a central server, and/or a human operator, and said distributed agents further utilizing said collected data to create and update a probabilistic model for a potential threat to the computer network.

19. A system as in claim **18**, further comprising a virus scanner that is updated by said pattern analyzed data from said distributed agents.

20. A system as in claim **18**, wherein said distributed agents are distributed across clients, servers and/or distributed central data warehouses of said computer network.

21. A system as in claim **18**, wherein distributed agents use threat detection, notification, agent updating, response and/or remediation protocols.

22. A system as in claim **18**, wherein at least one of said distributed agents provides defensive measures implementing redundant hardware, memory and communications links and neutralizes or isolates suspected threats to the computer network utilizing the defensive measures.

23. A system as in claim **18**, wherein said distributed agents implement adaptive learning techniques across a variety of system and network environments and are enabled by their capacity for interoperability between any heterogeneous protocols that are associated with a security system on the computer network.

24. A system as in claim **23**, wherein said interoperability between any heterogeneous protocols further enables said system to integrate with at least one other security system for purposes of implementing the system functions of passively collecting, monitoring, aggregating and pattern analyzing data as well as performing adaptive learning to detect the probability, classification, and/or nature of a threat to the computer network, developing notification thresholds for alerting other agents of potential threats to the computer network, and/or for developing appropriate countermeasures including defensive, remedial, reparative, and/or preventative functions.

25. A computer network threat and response system comprising a plurality of distributed agents that collect, monitor, aggregate and pattern analyze data representative of behavioral activities, content, and/or code in various locations across the computer network collected by respective distributed agents to identify similar patterns of suspicious activities indicative of an attack or threat to different portions of the computer network, determine from said pattern analysis whether a probability threshold of suspicious activity indicative of an attack or threat to said computer has been exceeded by said similar patterns of suspicious activities, and when said probability threshold has been exceeded, alert other agents, a central server, and/or a human operator; wherein attributes used for said pattern analysis include behavioral analysis of a hacker, code analysis, sequential event analysis, classification of the threat to the computer network, and/or textual and multimedia content features.

26. A network threat and response system comprising a plurality of distributed agents that collect, monitor, aggregate and pattern analyze data representative of behavioral activities, content, and/or code in various locations across a computer network collected by respective distributed agents to identify similar patterns of suspicious activities indicative of an attack or threat to different portions of the computer network, determine from said pattern analysis whether a probability threshold of suspicious activity indicative of an attack or threat to said computer network has been exceeded by said similar patterns of suspicious activities, and when said probability threshold has been exceeded, further provide notification to other agents on the computer network for purposes of defensive or counteroffensive responses, remedial repair, and/or recovery strategies.

27. A system as in claim **26**, wherein said notification includes a probability of likely threat to the computer network; subjection to other identified individuals, organizations, and/or nodes; degrees of potential danger; threat classification; and/or nature of said threat to the computer network.

28. A system as in claim **26**, wherein data associated with said notification is used to determine an origin of suspicious activity indicative of an attack or threat to the computer network for determining potential countermeasures to an identified threat.

29. A system as in claim **26**, wherein the said defensive or counteroffensive responses, remedial repair, and/or recovery strategies are customized to the overall conditions and circumstances characterizing a threat to the computer system and a threatened computer system.

30. A system as in claim **26**, wherein said defensive or counteroffensive responses include a honey pot trap.

31. A system as in claim **26**, wherein said defensive responses include implementing redundant hardware, memory, and/or communication links and neutralizing or isolating suspected threats to the computer system.

32. A system as in claim **26**, wherein said network threat detection and response system includes a network configuration based upon a distributed non-hierarchical or a hierarchical group of agents communicating with one another.

33. A distributed multi agent network security system comprising at least two agents that perform traffic monitoring and at least one of code analysis, content analysis, traffic analysis and creating activity models of various machines on a computer network and/or portions thereof representative of normal states and/or one or more abnormal states representative of suspicious activity indicative of an attack or threat to said computer network, each agent pattern analyzing collected data to identify similar patterns of suspicious activities indicative of an attack or threat to different portions of the computer network, determine from said pattern analysis whether a probability threshold of suspicious activity indicative of an attack or threat to said computer network has been exceeded by said similar patterns of suspicious activities, and when said probability threshold has been exceeded, further provide notification to other agents on the computer network.

34. A system as in claim **33**, wherein said at least one of code analysis, content analysis, traffic analysis and creating activity models is based upon the use of pattern analysis involving comparing data collected by each said agent to the data collected by another agent to identify similar patterns of suspicious activities indicative of an attack or threat to different portions of the computer network.

35. A system as in claim **33**, wherein said at least one of code analysis, content analysis, traffic analysis and creating activity models are used for purposes of threat detection, notification, agent updating, response and/or remediation protocols.

36. A system as in claim **33**, wherein said at least one of code analysis, content analysis, traffic analysis and creating activity models determine threat identification, threat classification, and/or probability of a threat to said computer network based upon at least one of observed characteristics, conditions/variables of an environment which a threat has encountered, data a threat has likely accessed, actions, events and/or countermeasures to which a threat has been exposed, code within which a threat has been embedded, dissemination of the threat through email, co-occurrence of identical or related patterns of code sequences in conjunction with suspicious behavior indicative of an attack or threat to said computer network, opening and/or modifying heterogeneous files, accessing a mail system's address folder, aggressive propagation of copies of the threat, recursively redundant actions, redundant messages, frequent or aggressive repetitive generation or obtaining of data files, propagation of inordinately voluminous or large files, redundant actions resulting in consumption of processing capacity, and/or modification or mutation of code and/or behavior.

* * * * *

<u>Exhibit C</u>

to

Complaint

for Patent Infringement

The ’470 Patent

US009503470B2

(12) United States Patent
Gertner et al.

(10) **Patent No.: US 9,503,470 B2**
(45) **Date of Patent: Nov. 22, 2016**

(54) **DISTRIBUTED AGENT BASED MODEL FOR SECURITY MONITORING AND RESPONSE**

(71) Applicant: **Fred Herz Patents, LLC**, Milton, WV (US)

(72) Inventors: **Yael Gertner**, Champaign, IL (US); **Frederick S. M. Herz**, Milton, WV (US); **Walter Paul Labys**, Fairfax, VA (US)

(73) Assignee: **Fred Herz Patents, LLC**, Milton, WV (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 291 days.

(21) Appl. No.: **14/043,567**

(22) Filed: **Oct. 1, 2013**

(65) **Prior Publication Data**

US 2014/0237599 A1 Aug. 21, 2014

Related U.S. Application Data

(63) Continuation-in-part of application No. 10/746,825, filed on Dec. 24, 2003, now Pat. No. 8,327,442, which is a continuation-in-part of application No. 10/693,149, filed on Oct. 23, 2003, now Pat. No. 8,046,835.

(60) Provisional application No. 61/708,304, filed on Oct. 1, 2012, provisional application No. 60/436,363, filed on Dec. 24, 2002.

(51) **Int. Cl.**
H04L 29/06 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 63/145*** (2013.01); ***H04L 63/1425*** (2013.01)

(58) **Field of Classification Search**
CPC H04L 63/145; H04L 63/1425
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,754,938	A	5/1998	Herz et al.	
5,754,939	A	5/1998	Herz et al.	
7,350,069	B2	3/2008	Herz et al.	
7,630,986	B1	12/2009	Herz et al.	
8,046,835	B2	10/2011	Herz	
8,327,442	B2	12/2012	Herz et al.	
8,490,197	B2	7/2013	Herz	
2002/0067832	A1 *	6/2002	Jablon	380/277
2003/0004688	A1 *	1/2003	Gupta et al.	702/188

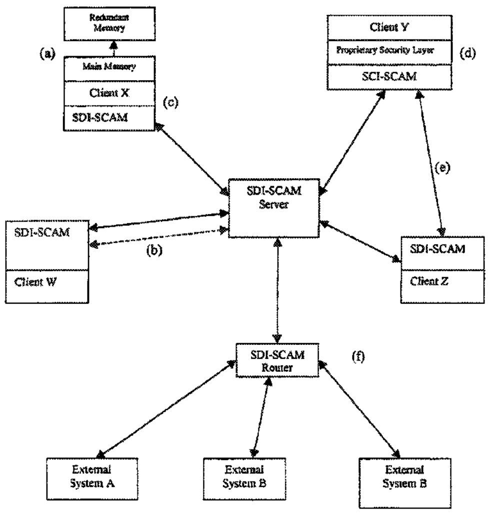
(Continued)

Primary Examiner — Jeffrey Pwu
Assistant Examiner — Thong Truong
(74) *Attorney, Agent, or Firm* — Baker & Hostetler LLP

(57) **ABSTRACT**

An architecture is provided for a widely distributed security system (SDI-SCAM) that protects computers at individual client locations, but which constantly pools and analyzes information gathered from machines across a network in order to quickly detect patterns consistent with intrusion or attack, singular or coordinated. When a novel method of attack has been detected, the system distributes warnings and potential countermeasures to each individual machine on the network. Such a warning may potentially include a probability distribution of the likelihood of an intrusion or attack as well as the relative probabilistic likelihood that such potential intrusion possesses certain characteristics or typologies or even strategic objectives in order to best recommend and/or distribute to each machine the most befitting countermeasure(s) given all presently known particular data and associated predicted probabilistic information regarding the prospective intrusion or attack. If any systems are adversely affected, methods for repairing the damage are shared and redistributed throughout the network.

22 Claims, 1 Drawing Sheet

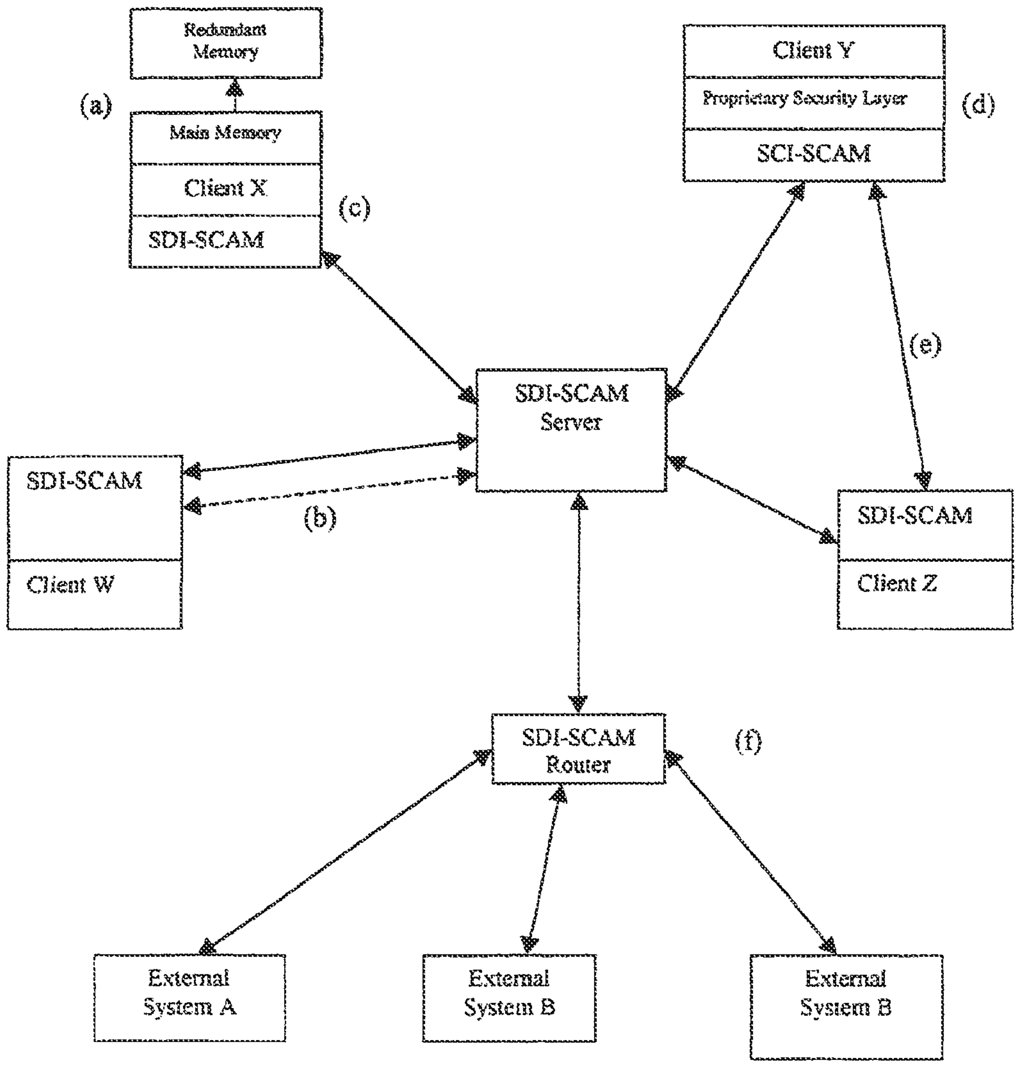


(56) **References Cited**

U.S. PATENT DOCUMENTS

2006/0069749	A1	3/2006	Herz et al.	
2006/0259970	A1 *	11/2006	Sheymov et al.	726/23
2008/0071578	A1	3/2008	Herz et al.	
2012/0131674	A1 *	5/2012	Wittenschlaeger	726/23

* cited by examiner


Redundant Memory
(a)
Main Memory
Client X
SDI-SCAM
(c)
Client Y
Proprietary Security Layer
(d)
SCI-SCAM
(e)
SDI-SCAM Server
SDI-SCAM
Client W
(b)
SDI-SCAM
Client Z
SDI-SCAM Router
(f)
External System A
External System B
External System B

DISTRIBUTED AGENT BASED MODEL FOR SECURITY MONITORING AND RESPONSE

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application claims benefit to U.S. Provisional Application No. 61/708,304 filed Oct. 1, 2012, which is a Continuation-in-Part of application Ser. No. 10/746,825, filed Dec. 24, 2003, now U.S. Pat. No. 8,327,442, which is, in turn, a Continuation-in-Part of application Ser. No. 10/693,149, filed Oct. 23, 2003, now U.S. Pat. No. 8,046,835, and further claims benefit of Provisional Patent Application 60/436,363, filed Dec. 24, 2002. These patent applications are incorporated herein by reference in their entireties.

BACKGROUND OF THE INVENTION

(1) Field of the Invention

The invention related to the field of security systems for computer networks.

(2) Description of Related Art

The United States and the rest of the world presents a target rich environment for a variety of cyber threats. Rogue hackers (e.g. Anonymous) have created significant disruptions. Criminal organizations have employed botnets and other strategies to engage in massive wealth transfers. And state-affiliated actors have penetrated a number of US entities, including the US Chamber of Commerce, Nortel, and others.

Particularly troubling is the fact that many of the activities have gone undetected for significant periods of time, suggesting that what we have seen is only the tip of the iceberg. Further, while much of this activity has been aimed at achieving political, financial, or diplomatic advantage, there is the troubling possibility of a coordinated attack taking out critical infrastructure in military, industrial, power generation, financial and other critical centers. If a previously prepared attack were coordinated with a significant conventional threat there could be global ramifications.

Per NSA Director Keith Alexander, perhaps a trillion dollars a year is being spent on cyber defense. This has not, however, bought a trillion dollars' worth of confidence in our cyber defenses.

It has proved difficult to bring threatened institutions up to currently recognized levels of best practice, i.e. use of secure passwords, single sign-on, least privilege, multiple firewalls, and so on. And it is far from clear that even current best practice is "best" enough.

Further, the large expenditures themselves represent a kind of failure. The current cyber-attacks represent a strongly asymmetric form of warfare. Individuals and small groups can create extraordinary DDOS attacks with only limited resources. The increasing complexity of software creates an exponential increase in potential points of attack. Increasing sophistication in hacks, e.g. self-assembling viruses, is providing new ways to exploit weakness. And the general sloppiness of the web—together with its use for more and more critical infrastructure—generates what one might euphemistically refer to as a "negative progress situation".

If attackers can spend small quantities of resource while generating large, expensive, slow, and relatively ineffectual responses, then even a successful response, if expensive enough, may mean a net strategic fail.

If we are to reverse this trend we will need approaches which are:

1. Adaptive
2. Autonomous
3. Automatic

They must be adaptive because the threats are, autonomous because the threats are high frequency and unceasing, and automatic because human response times are too great.

What is desired is something like an immune system for software, where even novel threats are recognized quickly and kick off a well-defined cascade of defensive and prophylactic measures, without conscious attention. Ideally the hyper-caffeinated hackers might spend weeks devising a new line of attack, only to see it flagged as abnormal and countered in milliseconds. The inventors have, ultimately, no objection to asymmetry, and would just like to see the sharp end of the asymmetry pointing in the other direction.

This is a non-trivial problem. Given the significance and difficulty of the problem, sound principles of portfolio management require exploration of a wide variety of approaches.

One approach to this problem is a distributed agent-based model for network security described in U.S. Pat. No. 8,046,835, and incorporated above by reference. As explained further below, this patent describes a distributed multi-agent system for real-time collection, monitoring, aggregation, and modeling of system and network operations, communications, internal and external access, coded execution functions, network and network resource conditions, as well as other assessable criteria. A Bayesian model is used to estimate likelihoods of various threat vectors. The model provides access to the reasoning behind its inferences. It may recommend or in some cases even implement responses to detected threats.

Since the time that the subject matter described in U.S. Pat. No. 8,046,835 was developed, cloud and other technologies have made tests of it significantly more feasible in the last few years. The basic idea is a large number of sensors, aggregators, and other agents monitor an at-risk system looking for anomalies. Bayesian analysis is used to estimate the probabilities that a particular pattern of activity is hostile. In the most sensitive cases, any variation from established baselines might be considered potentially hostile. Many of the agents could live in the secure cloud, keeping them from putting too much of a load on the defended system, letting their activity ramp up quickly when threat levels are higher, and keeping them themselves from being a target.

Since the individual agents are simple, once the overall architecture has been validated, tuning the system to respond to new threats and opportunities should be rapid, being often merely a matter of writing a small agent & telling the system to listen to it. This is analogous to the way "plug-in" modules are currently used to quickly augment the capabilities of browsers, word processors and the like. For example, if software certification authorities become generally available, certification agents could verify firmware and executables against published checksums, at random intervals or when an attack is suspected.

Calibration of the Bayesian model is of course key. Banks of reference systems (i.e. "honey pots")—some prepared clean, others prepared with known threats present—could be used to tune & validate the model. Use of cloud computing makes it economical to run thousands of tests simultaneously, ideally giving a relatively accurate way to judge under what conditions the model can be allowed to trigger an automated response and when verification by a human operator would be first required.

While the inventors are now focused on the cyber security problem, the original system was meant to be more general in application, with network security, financial, medical, and other applications. The agents can be used to:

1. Establish baselines
2. Identify variations
3. Generate smaller groups of agents to target specific threats (vaccines)
4. Initiate automated responses, as raising firewalls, switching to a spare machine not under attack, and so on
5. Serve as laboratories for developing counter-agents, verifying pre-deployment they will be effective against their intended target with a minimum of collateral damage.

Given that the development of such distributed agent-based models for security modeling and response is now feasible, the next step is develop some reference implementations, to test the ideas in practice. One approach is to work with existing vendors, i.e. Cogility or Blue Canopy, to see how this agent-based system can help automate their existing threat detection approaches. Another is to work with vanilla Linux systems, again with the goal of automating the synthesis of existing monitoring tools into an adaptive, autonomous, and automatic security monitoring and response system. The present invention addresses these needs in the art.

BRIEF SUMMARY OF THE INVENTION

An architecture is provided for a widely distributed security system (SDI-SCAM) that protects computers at individual client locations, but which constantly pools and analyzes information gathered from machines across a network in order to quickly detect patterns consistent with intrusion or attack, singular or coordinated. When a novel method of attack has been detected, the system distributes warnings and potential countermeasures to each individual machine on the network. In a preferred implementation, such a warning may potentially consist of a probability distribution of the likelihood of an intrusion or attack as well as the relative probabilistic likelihood that such potential intrusion possesses certain characteristics or typologies or even strategic objectives in order to best recommend and/or distribute to each machine the most befitting countermeasure(s) given all presently known particular data and associated predicted probabilistic information regarding the prospective intrusion or attack. If any systems are adversely affected, methods for repairing the damage are shared and redistributed throughout the network. The net impact of SDI-SCAM is that every machine on a network can benefit from security experience gained at any other point on the network. A high and uniform level of security is therefore assured to all systems attached to the network, and this security is updated in real-time.

In exemplary embodiments, systems and methods are provided for detecting the state of a computer network by providing a plurality of distributed agents disposed in the computer network to passively collect, monitor, and aggregate data representative of activities of respective nodes within said computer network. The aggregated data is analyzed to develop activity models based on collected data and representative of activities of the network in a normal state and activities of the computer network in an abnormal state as a result of intrusions, infections, scams and/or other suspicious activities in the computer network. The data analysis includes performing a pattern analysis on the col-

lected data to identify patterns in the collected data representative of suspicious activities. Counter-offensive measures are generated where unauthorized access to a program or file containing executable code results in the program or file disabling an operating system with all associated applications of a computer in the computer network until/unless the presumed attacker is able to prove to the machine owner/victim that the presumed attacker had been authorized to access the target data or machine provoking the said counter offensive measure.

The systems and methods of the exemplary embodiments also may aggregate and analyze data to develop a probabilistic likelihood of a threat to safe code, machines, servers, or individuals, and generate counter-offensive measures that are targeted to a given threat or attack based upon historical feedback from successes and failures of previous counter measures used in response to similar attacks and threats. An adaptive learning or adaptive rule base may be updated with the historical feedback. The systems and methods also may aggregate and analyze data pertaining to software within the network in order to enable detection and characterization of vulnerabilities and/or provide recommendations for remedial repair or revision. In addition, the systems and methods may generate a bogus target for invoking attack on the bogus target for purposes of achieving early detection of a system infection.

These and other characteristic features of the invention will become apparent to those skilled in the art from the following description.

BRIEF DESCRIPTION OF THE DRAWINGS

The FIGURE demonstrates some of the architectural features discussed, including (a) redundant memory within a given machine, (b) redundant connections between clients and servers, (c) SDI-SCAM installed as a primary security system, (d) SDI-SCAM piggybacking on an existing security system, (e) direct client-to-client agent communications, (f) on a router.

DETAILED DESCRIPTION OF THE INVENTION

The basic architectural approach for SDI-SCAM as described in U.S. Pat. No. 8,046,835 is that each node of a computer network is loaded with an agent capable both of ensuring security at the locality of the machine on which it is installed, and of communicating with other SDI-SCAM agents across the network. Because agent configurations are highly flexible, SDI-SCAM implementations can vary widely, running the spectrum from fully centralized (in which SDI-SCAM agents on client machines communicate uniquely with a centralized server dedicated to processing security-related information) to fully distributed (in which each client agent is given the ability to process security information locally, and information is shared on a peer-to-peer basis).

Basic Network Elements of SDI-SCAM

The preexisting elements of this network security system are the machines themselves. It is assumed that these systems, which act as the nodes of a network, consist of heterogeneous pieces of hardware running different sorts of operating systems. It may well be the case that various security layers will already be in place.

Additional Hardware

In preparation for the installation of SDI-SCAM across a network, it will often be desirable to upgrade existing machines with redundant hardware.

In a preferred embodiment, preexisting systems will be supplemented with redundant memory systems that persistently mirror the contents of the primary memory banks. When a computer's primary memory is corrupted (as can happen during a viral attack), it can be completed, cleared and reset with a pre-corruption image from the backup.

A further redundancy can be built into the network connections that link the local nodes to SDI-SCAM servers. For example, a computer that normally operates through land-based optic lines may be given an additional wireless connection through a satellite system.

An expensive, but preferred, architecture is to connect each SDI-SCAM agent through a fully isolated network that operates independently from the network on which the protected system resides. Thus, the SDI-SCAM agent will remain in contact with the security network even when the system it is supporting is under a sustained or unusually intense attack.

SDI-SCAM Agents

An agent is an entity that can be loaded onto any node(s) of a network, and which in this case is assigned responsibilities related to system security. Note that the construction of a given agent can vary widely, as it can be implemented through software, through hardware, through human interaction, or some combination thereof. In a preferred embodiment of SDI-SCAM, every machine linked into the system is loaded with an SDI-SCAM agent. Agent responsibilities include the following:

1) The collection of traffic data—among other things, each agent observes the packets being routed through its local system, observes every file transmission, monitors every user action, and logs every request for access.

2) The ability to communicate with other SDI-SCAM agents—each agent has the ability to communicate and exchange information with other agents (although the content of this information and the agents with which it is shared may be controlled, as will be discussed later). In normal use, a remote agent will send filtered traffic information downstream. When other agents detect potential security threats, warnings will pass upstream back to the remote agent.

3) The maintenance of various protections—On a continual basis, SDI-SCAM agents send and receive warnings and potential countermeasures relevant to whatever network risks are the most likely at a given time. For example, if a computer virus is detected at one node on the network, the local agent will immediately communicate a warning to all other agents in its contact neighborhood. If an attack is especially bad, the agent will have the ability to swap into the backup memory or contact other agents through alternative communications lines.

SDI-SCAM can operate either as a standalone security system, or as an additional layer that subsumes (and takes priority over, in cases of conflict) existing security protocols.

4) The ability to repair damage—Even after a node is known to have been attacked, the SDI-SCAM agent can be given access privileges such that it can aid the system administrator in controlling and repairing whatever damage has resulted.

5) The ability to scan collected data traffic for patterns consistent with threats—In many configurations, SDI-SCAM agents share their traffic information with a dedicated SDI-SCAM server capable of gathering and sifting through the entirety of the traffic data in order to detect

patterns consistent with a network attack, be it related to a hacker or to a new virus. Certain traffic events, which individually may be mistaken as simple anomalies, may become more apparent when the totality of a network's (or multiple networks) traffic is considered on a macro scale.

6) Notifying system administrators in the event of certain probabilistic attributes exceeding certain levels—The system's implementation of a Belief network (as herein disclosed) may also be used to determine under what overall conditions of probabilistically determined and descriptive variables it is advantageous to notify the system administrator. These variables can be based upon the predicted likelihood for the system to solve the problem, prevent certain types of problems, undesirable events and/or quantified degrees thereof from occurring or manual/or manually adaptive rules may prescribe threshold settings for some or all of these key variables. Among other situations, the system administrator may be notified or alerted in cases in which patterns detected may be only slightly suspicious according to the standard screening methodology, however, are consistent with SDI-Scam's best estimated simulation model from its distributed agent sources of how a threat might emerge, e.g., by mutation and re-emergence, e.g., after initially being defeated by SDI-Scam.

Meta-data associated with the accessor like a watermark that can also be embedded in code that contains digital credentials of the user, however, incorporates the use of "potentially" rogue, irresponsible, or destructive individuals as per the types of associated predictive attributes from criteria as disclosed in a presently preferred embodiment. The code cannot be tampered with without interrupting the watermark. A more general term for this "invisible" code sequence, which appears random to a would-be interceptor, is "embedded code". Typically, the embedding is done in a much larger nonsense message to apparently random patterns (in as much as the application code would already be encrypted) and this nonsense message content may not be required. Also, it can be associated with functionally defined portions of the code, which pre-approve certain behaviors. The system could also be based upon willingness of the accessor and/or code which s/he writes to statistically pseudonymize and profile the user with that of the patterns/types, etc. of code s/he has written in the past, thus predicting even without explicit identification who is the likely author and what s/he is like, i.e., what is the statistical probability distribution of the individual to each of a variety of previously known identities based upon code morphological characteristics, functional behavioral features, human behavioral features (e.g., if it is accompanied by a human attack). Pseudonyms and resolution credentials may be useful to authenticate the basic intent and MO of the author of the code while use of cryptographically interoperable pseudonyms, i.e., multiple unique but single identity aliases which are linkable to that single author only by SDI-SCAM for its security analytical purposes and under prescribed conditions (as data disclosure policies) as dictated by that author. Pseudonyms may be used to insure the same level of anonymity of the author as uncredentialed code. This approach could, of course, either be implemented as a local protocol (i.e., existing applications, application updates and new applications could all possess these credentials verifying/certifying that the present code was written by an individual who has been certified by a trusted certification authority as being non-malicious). This approach and the above pseudonym based identity protection scheme, while applied in this case to the application of software security are disclosed in detail for the application of identity protection, data privacy and security from rogue individuals interacting on communication networks such as the Internet. These relevantly related techniques are well described in the parent case as well as in U.S. Pat. No. 5,754,938, entitled "Pseudonymous Server for System for Customized Electronic Identification of Desirable Objects".

Within a typical context, this type of code certification should be impervious to a "man in the middle" attack. Such embedded messages (or in a similar cryptographic variation, "fingerprinting") are inherently effective for the security application proposed inasmuch as any rogue code which a system attacker would attempt to insert into a certified application or communication or other communication containing executable code would contain within its sequences continuous portions which do not contain the embedded credential-based sequences. Likewise, in case the would-be man in the middle attempted to remove certain data, (e.g., credentials or functional application code) the fingerprinting technique would recognize the specific extracted code segments. This exact same problem can be solved alternatively another way in which the primary objective is to transmit data containing a message the existence of which is not possible to be detected by a would-be "man in the middle" attacker. In the example approach in which a content bearing message is embedded or fingerprinted into the application code (or less desirably in an associated larger message), the message can only be identified by the recipient (the local SDI-SCAM agent) who may also be similarly hidden or "steganographed" as with the originally sent message (in order to verify receipt of the message by the authenticated recipient. There may exist in this content bearing message a variety of useful credentials incorporated therein including but not limited to credentials approving both authenticity, untampered state and authentication of the sender and/or author as well as proof of certified "good intent" on the part of the code author. The technique for insuring that the embedded sequences are completely undetectable, while at the same time being diffusely spread throughout the code is typically performed by using encryption techniques (e.g., pseudo-random sequences) to determine the positions of the sequence bits within the remaining code in order to thus pass a message to the recipient (the local SDI-SCAM agent) containing the credentials and potentially the message of the coordinates of the associated meaningful sequences, such that all of these content bearing sequences appear among the remaining code as random noise, including the portion of the message containing the encrypted coordinate data of which coordinate bits possessing the totality of the embedded or fingerprinted message can be found within the application. Alternatively, this message containing the coordinate locations of where to find the meaningful bits containing the content bearing message may be embedded within a larger message which itself appears to consist entirely of noise (which in and of itself lends the security of the embedded or fingerprinted message contained therein). The primary hurdle in this case is to enable the recipient to be privy to certain data, which is not known to a would-be "man in the middle" attacker namely where to look for the message, i.e., the coordinates of the meaningful data constructing the message. This "shared secret" between the sender and the receiver could be conveyed to each party initially by a (one time) physical distribution (e.g., contained within an application if it is physically distributed, such as on a disk, or visa vie the OS or CPU, etc. In one variation in which the dissemination of this message needs to be performed on a network wide level (or group level), the shared secrets may be physically distributed, once to all parties in a group and, subsequently, all parties would be able to instantly initiate communications with the security guarantees achievable through the presently proposed methodology.

Finally, it will be sufficiently obvious to one skilled in the art that the presently proposed methodology has numerous potential applications in cryptography and data security and thus the means for distributing data coordinates to a recipient of a steganographed message for conveying (and if desired reciprocally confirming) a message is in no way limited to messages, containing credentials and authentication certificates about an author and/or sender. For example, the present technique could be very prudently employed as a means to distribute and replenish shared set keys within the context of U.S. Pat. No. 7,350,069. It may also protect against man in the middle attacks against distribution of private keys in Pki protocols.

SDI-SCAM Network

There are multiple network morphologies possible. Major configurations include the following:

1) Local network: SDI-SCAM enabled machines may form a local network, such as a LAN or WAN. Gateways to external networks (such as the Internet) can be fully controlled through SDI-SCAM enabled routers.

2) Open network: On the other hand, SDI-SCAM enabled machines can be connected directly to outside systems (such as a desktop system connecting through a generic ISP), but which maintain communications with a chosen neighborhood of other SDI-SCAM enabled machines.

3) Centrally organized networks—In this configuration, thinner SDI-SCAM agents are placed on individual nodes; these agents continue to be responsible for direct security and repair, but transmit gathered traffic information to central SDI-SCAM servers containing dedicated hardware and software capable of swift and very in-depth analysis of the gathered information.

4) Distributed networks: In this configuration, each SDI-SCAM agent shares the responsibility for traffic data analysis and the generation of preventative measures with other agents. A peer-to-peer morphology would work well in this case.

Inter-Agent Communications

Although there is clearly a benefit for agents to fully pool all information, it may be desirable to control both the content shared and the partners with which a particular agent is allowed to interact. These parameters can be set at the local level according to users' preferences.

SDI-SCAM agents may in fact negotiate with each other depending on the value and sensitivity of particular information, as well as the value of any likely synergies between them. Multiple agents may meet in virtual information sharing marketplaces.

Another level of security can be gained through the exchange of obfuscated, but still valuable, information. Such randomized aggregates would allow systems to share fundamentals without revealing details of their particular data (for example, agents could share times of attempted log-ins without revealing the associated user ids and failed passwords).

In more complex realizations of this system, associated groups of agents may form coalitions, with information shared freely internally, but shared with conditions externally.

A further feature is that communications between agents need not be perfectly symmetric—in other words, different agents may send and receive different sorts of information. This might apply, for example, to a centrally organized SDI-SCAM network: outlying agents would have no need to transmit detailed traffic data to each other, but would rather transmit it directly to a central server. The central server might communicate with other central servers, in which case it would transmit high-level information relevant to the processing of the entirety of the traffic data; on the other hand, when communicating with outlying nodes, the central server might only transmit simple virus protection instructions and metrics which are substantially devoid of any data which suggests what types of information, attacker strategies or applications are running on other nodes on the system which are outside of the network of nodes and which are currently trusted by the nodes from which the centrally collected and processed data had been acquired.

Furthermore, there may be an additional or alternative approach to guaranteeing absolute data security at a local network or machine level while enabling maximal or complete harnessing of all of the statistical knowledge, which is present across the entirety of the network. In this approach it may be possible to operate SDI-SCAM or certain particularly sensitive portions of it with its multiple agent architecture as a singular trusted, yet distributed multi-agent system. In this variation, all of the locally performed or assigned agent functions are assumed to contain sensitive data belonging to external third parties and thus all processing activities, data communications with other agents or the central SDI-SCAM server occurs within a secure trusted and untamperable environment such that the only knowledge ultimately accessible by any given agents, associated local server or network on which it physically resides may be the collection of executed functions which are performed by the local agent on behalf of the SDI-SCAM to protect the local system as herein disclosed.

The order and way in which agents communicate with each other may be highly conditioned on the particular nature of a given system. Criteria include (but are not limited to) the following:

overall vulnerability of a system;

importance of the system to the integrity or functioning of a network;

sensitivity and value of the data stored on a system;

probability that the system has already been compromised or damaged;

characteristics of the network traffic going to and coming from the system;

overall importance of a system to a potential or identified hacker or specific system subcomponent.

This may dynamically change from moment to moment and is predicated by a probabilistic estimate determination variable of the intruder, whether autonomous or human and/or by human expert based estimates who are ideally familiar with local competition (or enemies) and broad knowledge of what types of knowledge on the system would be most of interest to which other entities or individuals and for what reason. If an individual is specifically identified this statistical model may further borrow and integrate techniques disclosed in co-pending U.S. patent application Ser. No. 11/091,263, filed Mar. 26, 2007.

Updates and communications between agents (termed "polling") may be based on schedules or on circumstances. For example, a remote agent may be updated with new antiviral software once a month; however, if any other node on the network is attacked, the schedule is suspended and an immediate update is performed. Certainly even if an attack which, for example, has only begun to occur or which has not even positively been confirmed as yet, triggers SDI-SCAM's system alert feature, other nodes on the network most preferentially/urgently those which are physically proximal or in other ways similar may also be put on alert status and SDI-SCAM's repertoire of protective features may be triggered so as to begin operating at a heightened level of defensive activity. As indicated, there may be a range of different system defense levels corresponding to a decreased probabilistic likelihood of a threat and the likely severity thereof should this threat exist. Local system administrators are notified appropriately as well. Determining the likelihood that a threat upon a particular node or network will also be carried out against any other given node can be predicted by such variables as commonalities at an organizational or strategic level, data communication occurring there between, commonalities in the existing or perceived data on applications contained or functional objectives achieved upon that node, presume interest level that a potential intruder of the attacked node or network may also have with the other node, etc.

Polling priority may be based on calculated likelihoods: for example, if various factors indicate that the probability is high that a remote node has been infected by a particular type of virus, the central server may be put into immediate communication. Polling priority will also depend on the nature of the nodes and the way in which their agents have been seen to communicate. U.S. Pat. No. 5,754,939, entitled "System for Generation of User Profiles for a System for Customized Electronic Identification of Desirables Objects" may be used as the basis for optimizing the way in which polling is performed.

Illustration

FIG. **1** provides an illustration of some of the configurations discussed here.

Analytics

Given the number of different security objectives, as well as the number and diversity of possible agents and network configurations, a fairly broad range of analytical tools are employed by SDI-SCAM. They include, but are not limited to, the following major categories of analysis:

Methods to Detect and Classify Direct Intrusions

Direct intrusions are attempts by unauthorized entities to enter a particular system, either over a network or through local terminals. These can range from fairly unsophisticated attacks (for example, teenage "script kiddies" using standard public domain software to scan for open ports across a list of target IP addresses), to extremely skillful attacks that are focused on a very particular target (as might happen during corporate espionage).

Since SDI-SCAM agents are able to dynamically monitor and analyze as well as control all in-going and out-going traffic, they are in a good position to detect and counteract such attacks.

1) Attack Patterns Consistent with Previously-Observed Patterns Across the SDI-SCAM Distributed System.

Each SDI-SCAM agent has access to a shared database that contains the signature patterns of previously observed (as well as verified) attacks. The likelihood of these events having been actual attacks may be probabilistically estimated so as to optimize the precision of SDI-SCAM detection/diagnosis as well as countermeasure deployment system modules. Such patterns might include the use of a particular password list, log-ins at particular time intervals or frequencies or times, log-ins from suspect IPs, (and/or combinations thereof) constitute a few of the straightforward examples.

If such a pattern is detected, the resident SDI-SCAM agent may opt to deny all entry to the IP of the incoming log attempts, or it may opt for a more sophisticated defense, such as opening a "honey pot" trap, a virtual space that simulates the environment of the system that is being protected. The hacker, believing that he has actually broken into the system, can then be monitored by SDI-SCAM, as his behavior might give clues to his location, identity, and motives and incriminatory evidence, if desired. Assuming the hacker has learned (or possesses) enough knowledge about the system to detect "honey pot" traps it is advantageous and precocious to possess at least equivalent knowledge regarding SDI-SCAM to possess at least equivalent knowledge regarding its own environment and to be able to enable the system administrator access to that knowledge as well as (via SDI-SCAM) knowledge known or suspected to exist within a probabilistic context regarding the hacker or threat and its strategy and/or this knowledge may be acted upon appropriately by SDI-SCAM in automatic mode. Invariably all counter measures (such as honey pot traps) used by SDI-SCAM can be used to the advantage of the hacker if s/he is aware of the strategy of SDI-SCAM to monitor, model, locate in order to ultimately catch him/her.

2) Utilizing Data Modeling to Adaptively Learn and Recommend Appropriate Countermeasures

Implementation of practically viable automated countermeasure scrutinization and recommendation scheme is quite achievable:

a. If the conditions/parameter triggers are simple and unambiguous, and

b. If the system administrator is notified and able to intervene while exploiting the system's analytical knowledge and system-generated recommendations and scrutinies by the system on behalf of his/her chosen response decision.

In the ideal scenario, because rogue attacks are capable of performing increasingly effectively against system security protections (in addition to being more sophisticated and expeditious) and especially with regards to leveraging the system's own abundantly capable resources, it may be ideal as a complementary measure to building redundancy into the system resources in the interest of expediency of decrypting a counter measure, to also immediately respond in automatic mode, then solicit the active, albeit system-guided intervention of the system administrator whereby more significant decisions can be perhaps more confidently and prudently executed (e.g., whether or not to delete potentially corrupted files/portions of system data at the server or network level), whether to guarantee a certain portion of the network but allow certain essential functions to continue for the time being without code exchange, whether or not to attempt to infect the hacker's machine (or analysis code into the virus itself) which may provide additional detailed information as well, etc.

3) Novel Attacks

In some cases, attacks will follow completely new or novel patterns.

Such attacks can be detected in different ways. One solution is to configure a Bayesian network to constantly gauge the probability of an ongoing attack by monitoring network traffic activity (this configuration can be done by human experts and/or through machine learning techniques). A variety of factors can be extracted from the network traffic across all SDI-SCAM agents in the local network—for example, the number of failed log-ins, the identities and IP addresses of those users attempting to log in, the importance, sensitivity or "value" (more specifically "perceived value") of particular target files or contents potential adversarial entity or prospective hacker, etc. These factors are fed into ongoing probability calculations, which may trigger a system-wide warning if a certain threshold is surpassed.

Keystroke monitoring virus must be mentioned since it is impervious to NORTON™, etc.

For example, suppose a ring of corporate spies tries to hit a company's network simultaneously. SDI-SCAM agents across the network will report the use of unauthorized passwords originating from the same IP or IPs to which associations have been constructed via SDI-SCAM based upon historical statistics if the probabilistic likelihood of such events occurring independently might be so unlikely that the Bayesian network would immediately increase its estimate of an ongoing attack.

4) Attack Warnings

Note that in all cases, when an attack is suspected the resident SDI-SCAM agent will immediately alert all the other SDI-SCAM agents in its network neighborhood, sharing all traffic information relevant to the on-going security issue. Such warnings will include information related to the particular nature of the problem, in particular the probability and nature of the threat (for example, communication with an unsecure system, access by an authorized user, reception of potentially infected files, etc.).

When an on-going attack is announced, SDI-SCAM agents receiving this information may opt to increase the security levels of their own systems. For example, users may be required to telephone at the time of their log-in to verify location (through caller ID) and voiceprint.

Methods to Detect and Classify Viruses or "Trojan Horses"

Origins, possible paths of transmission across sites, etc. types of files (e.g., particularly vulnerable or vulnerable origin site), may be analyzed to provide ideas as to how to use this data to make a vulnerable application, Trojan horse attempt impervious, make rogueness, crypto query, even rewrite code.

Another vector of attack is through viruses (which are often unauthorized and malicious programs attached to files, email, or documents) and Trojan horses (seemingly innocuous programs that contain hidden programming capable of causing damage).

Code Analysis

The conventional viral detection methodology is to scan the code (in the case of executable modules) and macros (in the case of smart documents, such as those generated by Microsoft WORD™) for patterns that have previously been associated with viruses or malicious programming

SDI-SCAM maintains up-to-date records of all known viruses and checks all incoming files (and periodically, all stored files) against these records. A match indicates that a file is potentially infected—the user is alerted of the danger and automatic defensive measures may be set into motion.

Behavioral Analysis

SDI-SCAM monitors all processes for behavior consistent with viral infection. For example, a program that is observed to open and modify a wide range of heterogeneous files, which accesses the mail system's address folder, which aggressively propagates copies of itself, which engages in recursively redundant actions whose objective is designed to achieve no useful purposes or frequently which aggressively/repetitively generates or obtains data files in order to propagate inordinately voluminous and/or large files (possibly including itself) resulting in bursts of traffic (thus overloading valuable network transmission capacity), which performs similar recursively redundant actions resulting in consumption and overloading of valuable processing capacity, which modifies or mutates its own code (and/or behavior), or which opens unexpected communication ports with outside entities will be flagged as a potential threat. Unquestionably, SDI-SCAM's highly distributed data traffic monitoring and behavior and code analysis facilities as a combined approach give it a marked and compelling advantage in rapidly analyzing those behavioral patterns and characteristics most commonly associated with a rogue code such as viruses, Trojan horses, worms, etc. whose tell-tale signs could not be identified nearly as expeditiously as that of SDI-SCAM's distributed agent monitoring architecture. Such commonly occurring signatures which SDI-SCAM's distributed Bayesian methodology is particularly well suited includes those patterns of self-replication and dissemination through address books, email, web browsing sessions, etc., as well as the co-occurrence of identical or related patterns of behavior and code sequences in conjunction with these replicating and self-propagating patterns as observed only on a network level. Certainly part of this behavioral analysis may encompass attempts by SDI-SCAM to classify the identity or type of virus based upon all of the above observed characteristics as well as attempting to extrapolate its high level objectives and associated executable rule sets based upon its behavioral patterns associated with the conditions/variables of the environment which it has encountered, the data which it has likely accessed, the actions, events and countermeasures to which it has been exposed, the code within which it has likely been embedded, etc.

Although it may be difficult to delineate rogue from innocuous code it is certainly within the scope of capabilities of SDI-SCAM to utilize all of the available data, both behavioral and code sequences, in order to attempt to reverse engineer the code for the purposes of both predicting its future behavior, likely past behavior and high level objectives. For example, SDI-SCAM could replicate the code inside of an environment which is quarantined from the network, but which is a replica of the network or a portion thereof. SCI-SCAM could then monitor how the code behaves in this simulated environment to the actual one as well as observing its response to targeted stimuli, which may, for example, provide opportune conditions for the most likely rogue actions to be performed. This analytical procedure may be performed in response to a predicting statistical model (designed to predict the code's behavior) when a decision tree could be used to dynamically select the set of functions to be executed which based upon the same model are correlated and then predicted to elucidate responses on which are the most optimally revealing, reveal the most revealing which is needed to complete the construction of this data model for the codes for being able to predict the code's behavior across a wide array of conditions, actions, software and data to which it may ultimately become exposed within the entirety of network(s). In depth analysis of potentially suspicious code although challenging as it may be could potentially provide system level insights into how to best respond to the potential threat and if mandatory the nature and aggressiveness of countermeasures to be taken or recommended to the appropriate human system security counterpart.

The user will be alerted, and if he confirms that the program is operating outside of expected parameters, or if the user does not recognize the program, it is taken offline until it can be examined in detail by an expert.

Dead-Ringer Analysis

Although not currently a threat, it is likely that infectious programs will be able to simulate the behavior of human users. A suite of behavioral response tests can be developed to detect and counteract such entities, e.g., a probabilistic model based upon other previous threats in the statistically similar characteristics (including behavioral characteristics and certainly those determined to be the most likely to be the

same). Queries which may be required of the "user" to be answered correctly or to perform a task (e.g., compose a block of text on the basis of a query) in order to proceed could be solicited of the user which are crafted such that an emulating virus would likely fail such query procedure. Moreover, Natural Language Processing methods can be used to analyze outgoing text for irregularities consistent with a non-human origin. It is possible that in a similar fashion, that, in theory very smart emulations of existing code could be manually or even automatically on the fly created which emulates in many respects existing "good code", but which actually is designed for malicious objectives or, for example to take over control of the good code or replace it with the rogue version. As additional attributes of the system, the system may determine probability and degree of ill motive of individuals of most likely suspicion (if such suspicion is high enough to be of reasonable concern). Typically, common suspicion of particular individuals can be linked to unscrupulous employees (present or former), disgruntled employees, disgruntled spouses of key persons/owners (e.g., changing files, information release, etc.) to embarrass or defame the person or to feign a verbal or tactical attack on a friend, associate or colleague. Such "suspects" could also include trusted partners who may be confided with knowledge of the existence of unique information which could be of interest directly or could even help or strengthen that party in its business position with its "trusted" business partner.

Control of Triggers

If the probability of an infection is deemed to be high, SDI-SCAM may control the generation of events that could potentially trigger the reaction of a resident virus. For example, if a bank suspects that a corporate virus has infected the system, all transactions may be suspended until the virus is cleared. Otherwise, the action of a user requesting an on-line transaction (thereby releasing his personal password to the system) may trigger the virus into capturing and re-transmitting his personal information.

Tracing Threats Back to their Original Source

In traditional system security techniques this objective is highly desirable and yet extremely difficult. Nonetheless, SDI-SCAM's functional features lend themselves quite well to the design of certain particular types of applications, which can be useful in addressing this particular problem. For example, the following example applications may be herein considered:

1. "Infecting" the Hacker's Machine (or the Virus) with a Virus, which Logs and/or Conveys Back to the SDI-SCAM Agent the Location, Behavior, Files Infected as Well as all IP Addresses of the Machines in which these Files Reside.

This approach is likely to work provided that the implanted virus by SDI-SCAM is not recognized by standard virus scanning software or other IDS systems and assuming that the receiving machine is not programmed to block any outgoing messages. Thus, the success would be determined in part by the effectiveness of the virus to take control of the adversary's (or rogue virus containing) machine. This type of direct analysis will both enable preemptive alerts of exactly where the virus may be spreading to other machines and/or networks as well as provide valuable statistically confident data as to the function, behavior, data or code affinities and behavior in response to infection of the same as well as epidemiological characteristics which could be extremely valuable as to anticipatory determination and qualification of the associated threat on other machines, as well as the most appropriate countermeasure each local agent should implement or receive in response. Certainly, this approach could be useful for viruses, which possess particular rapidly proliferating characteristics, rapid infliction of destructive behavior. For example, one could imagine the behavior of more sophisticated viruses which might proliferate themselves as redundant messages so as to rapidly overwhelm network capacity and/or memory processing and/or implement parallel strategies.

This approach could also enable SDI-SCAM to model not only future epidemiological characteristics of rogue software but also that of post epidemiological behavior (which machines or networks were likely to have been infective previously based upon presently known epidemiological characteristics) and the devices/networks which are known to be and probabilistically suspected of being infected by the same virus (or mutated variant thereof). Certainly reconstruction past, present and future behavior in this regard could be relatively easy to perform for worms that may require access to ISP server logs for other variations which may use email and web server connections as a medium of transmission. A protocol also may allow for the existence of a latent tracking virus to reside within all machines which can be, in the case of significant probability of a threat in and among a network community or otherwise "group" an excessive probability of a threat, the tracking virus may be remotely activated by a multi-casted activation message originating form a core (or root) server.

2. Use of SDI-SCAM Architecture for Application Level Security

It will be increasingly important in the future for many of the functions of SDI-SCAM as implemented within the context of its presently disclosed distributed statistical analytics to be implemented not only at the level of a distributed network security system but also at the individual application level. That is to say that SDI-SCAM agents could, in addition to the above described system level implementations, also implement their various functions for data collection, analysis, and countermeasures at the application level as well both to implement other application level security protocols as well as incorporate into the statistical analytical scheme probabilistic attributes regarding the behavior functions, etc., of such rogue code within the context of the particular relevant applications in need of protection, albeit using the same distributed adaptive modeling and countermeasure response protocols described herein in comprehensive fashion.

Methods to Detect Tampered Files (Semantics and Content)

It is sometimes the case that intruders, rather than destroying or removing files, will simply alter them in potentially malicious ways. For example, students may attempt to hack into their school system in order to change grades, or a more advanced hacker may attempt to break into a bank to fraudulently increase the balance in his account, into tax or criminal record databases in order to change tax liabilities, records of property ownership or criminal records, into professional board's databases in order to change licensure status. Similar tampering may occur to files whose contents may relate to the hacker (e.g., employee files of present or past employers). Malicious code may, in theory, perform all of the functions that a human may perform, perhaps, however, potentially even more unobtrusively and elusively in that it may be more difficult to trace and flag than a human if the code is very small, robust and capable of focused but sophisticated emulations of legitimate applications and users.

In addition to the above suggested techniques for use in tampering detection and ultimately prevention (or even

tracing the origins of tampering attempts), there are other straightforward IDS-based approaches by which such attempts could be countered (and could even complement the above safeguarding scheme, for example, in terms of being a default detection scheme and/or in corroboration of the presumed integrity of credentialed individuals). Thus, the following IDS-based alternative technical approach is also provided as well. The local SDI-SCAM agent maintains logs that detail the general characteristics (size, word count, hash code) of all documents on the system. The time and circumstances of any changes are cross-checked against average traffic patterns for that particular system. Hence, school records altered at 3 am (in a district where all school secretaries worked strictly from 9 am to 5 pm) may be flagged as potential objects of tampering.

Tampered files will sometimes show a marked change in writing style or technique. Natural Language Programming (NLP) techniques may be used to detect such changes. Certainly in the event of these suspicious activities and other conditions, it may be advantageous to retain not only the associated statistical data (as the SDI-SCAM does automatically) but also details regarding the events. This could, for example, be later analyzed by humans to compare with other similar suspicious patterns also captured in detail in order to attempt to identify patterns, more subjective signatures, or hall marks which may not be able to be performed automatically (such data may also be useful for potential legal evidence).

Methods to Detect and Classify Untruthful Commercial Messages

Untruthful messages represent a more traditional kind of deception—the technology of the delivery is not damaging, rather, the content of the message itself is untruthful and may prove harmful if taken at face value by the receiver. A good example of this is the "Nigerian Scam," a widely disseminated email that purports to be authentic, asking the receiver to give the sender access to an American bank account in exchange for great financial reward. The result, of course, is that the receiver ends up being defrauded of large amounts of money.

1) Cross-Checking Content Against Known Hoax Documents

SDI-SCAM maintains a database of questionable messages and uses natural language programming-based techniques to compare incoming messages with previously logged deceptions. Thus, when a suspicious message is detected, the receiver may be sent a secure attachment by SDI-SCAM with an email stating that there is a high probability that the mail is untruthful, and indicating pointers to web pages that discuss that particular deception. If a user is nonetheless deceived by such a message, the local SDI-SCAM agent may be alerted. It will transmit the text of the novel message to a security database, allowing every other SDI-SCAM in that network neighborhood to be aware of the danger. In such a case, the agents may retroactively warn users by scanning old emails and alerting receivers of possible deception.

Certainly in such an event, autonomously implemented counter measures may also be performed if appropriate as a defensive or evasive action or deterrent, e.g., if a pass code was inadvertently sent out (and it was not blocked by the system) the pass code could be automatically changed or temporarily frozen or if a personal bank account or credit card number were sent out in a suspected inappropriate context (again assuming it was not blocked at the source by the system), the account could be automatically temporarily frozen and the number changed or (for example) the account automatically set up as a honey pot trap to acquire just enough information about who the suspect entity is in order to catch him in an inappropriate act of fraud or deception.

2) Predicting Possible Hoax in Novel Message

In cases where a message is not closely correlated with known hoaxes, it is still possible to analyze (using natural language processing techniques that are currently well known to the art) the content of the message and flag any suspicious content:

the content of the message can be cross-checked against recent news stories discussing hoaxes; and,

the names and return email addresses of the incoming mail may be checked against those of known hoaxsters.

Automated semantic analysis of the message may be performed for language consistent with persuasion or appeal to greed (or other weaknesses). This analysis is performed on the basis of adaptive rules which may be updated with feedback.

The identity and personal profile of the receiver may be correlated with the characteristics of known victim groups. For example, messages sent to rich elderly individuals may be given additional scrutiny.

The purported identity of the sender can be checked against the path of the email. For example, a message claiming to be from the IRS should trace back to an official government system.

A probabilistic assessment of the likelihood that the sender is fraudulent may be performed through a modified version of the system described in co-pending U.S. patent application Ser. No. 11/091,263, filed Mar. 26, 2007, in which the system's probabilistic determination of predictive attributes relevant to an association with fraudulent, unscrupulous or disruptive behavior (in an on-line context) is performed—of course, the sender if self-identified may also be fraudulent. The on-line sender just prior to the first receiving node on the system may also be analyzed which is a reasonably reliable tracking means if SDI-SCAM is a ubiquitous protocol (e.g., for patterns of being the origination node for previous problematic messages and/or the techniques disclosed in the same co-pending patent application), whereby the system may probabilistically predict the suspicion level of an individual(s) or organization(s) associated with that sender as being linked to other scams and/or other illegitimate or questionable activities. Related techniques may use other advanced customized semantic analysis and/or adaptive rule based/statistical techniques (preferably in combination) in order to estimate the degree of potential harmfulness of the content.

Because SDI-SCAM's distributed agent and sensor network perform consistent detailed surveillance capable of monitoring across a diversity of different heuristics, it is possible to monitor for early patterns which are indicative of an attack which may be quite sophisticated either of a large-scale nature or which targets a specific type of system and therefore which may tend to go undetected (e.g. visa vie its hardware, operating system or possibly software) but may be destructive to very important systems like control systems for critical infrastructure. These detectable patterns may be based upon experience of the agents (to previous known threats) and unknown threats e.g. detecting patterns which are not normal but not definitively linked to threats of a previously known nature. Certainly fuzzy probabilities analyzing of combinations of these condition states will be able to fine tune triggers for potential alert states which are matched appropriately e.g. balancing potential threat to the possibility of variations of normality to the real degree of threat and given the degree of importance of early detection and response given the overall importance of the systems at hand and the predicted virulence of the threat that the detected pattern presages.

Thus with regards to specialized systems and networks, certain specialized patterns corresponding to the type of end result that the attacker may like to achieve may be preprogrammed into the system even if such novel attack had not occurred previously. If there were infections of specific types of operating systems or software or hardware which also corresponded to certain of these components that existed in critical infrastructure it would be cause for concern that such a threat (e.g. a worm) may in fact already exist for purposes of attempting to target such critical infrastructure. This is one reason that redundant components of this type which may be placed on the network or even assembled into analogous system architectures may be useful in preemptively becoming targeted (as "bogus targets") for purposes of early detection and characterization of the threat before it actually targets its intended critical system. This is just one example of many techniques and strategies which can be used by SDI-SCAM to complement its ability to utilize its distributed sensors to identify and characterize potential threats either at the very onset or even before they occur or certainly as they are emerging and beginning to spread. Again the nature of the automated notification, additional defenses and/or remedial counter responses may be both targeted to the threat, to the targeted and/or vulnerable systems as well as to those where SDI-SCAM is able to predict the spread is most likely to occur, e.g., based upon commonalities of various sorts such as communication links, business/commercial affiliations or social profiles, etc. On traditional networks in monitoring for potential signs of large-scale attacks experiential knowledge must be of course heavily weighted. There are a growing variety of and new variations of such threats. However they tend to fall into general categories for which significant experiential data can be accrued.

The content may be corroborated with the content of known and trusted documents, e.g., through the use of content matching techniques. More elaborate extensions of this approach may include more advanced semantic analyses of the subject content with its credible and updated/current matching counterparts whose algorithms are custom configured to confirm (or alternatively flag) or assess the probabilistically estimated "truthfulness" of contents (where "truthfulness" may be reassured according to "confirmed with credible source" as well as scalar measures of degree of likelihood of untruthfulness if the source is unconfirmed or, for example, exhibits semantically suspicious inconsistencies with itself, with credible sources or other patterns which are consistent with fraudulent or deceptive material).

The system may also detect suspicious content, for example, if its appearance co-occurs in the same message with rogue code (for example) is co-located (in the same portion of content) as a macrovirus.

Methods to Repair Post-Attack Damage

In some cases, despite the security, a system in an SDI-SCAM network may be damaged by an attack. If the attack is completely novel, a human expert may be called in to fully analyze the situation and develop appropriate repair protocols. These can then be sent to a central SDI-SCAM damage-control database for use in future situations. In this way capturing as much data and statistical information regarding the attack and its historical counterpart is valuable both as analysis data for the human or to enable the system to construct its own optimal repair protocol.

If an attack method is not novel, the local SDI-SCAM system may access this same damage repair database for solutions to the local problem. Among the remedies to damage from an attack: users are alerted, suspicious files are deleted, backup files are loaded, and current memory is completely cleared and reloaded with an image from a pre-attack state.

ADDITIONAL EMBODIMENTS

The inventors further propose a scheme in which a Server provides security and validity information about free online software that can be downloaded off the web. The Server keeps a score that is made up of three components. The first component is computed based on the security of the software itself. The second component is computed based on the experience users have with the program. The third component is based on the reputation of the programmer that created the program.

Component 1: The Security of the Program

In order to make sure that the program does not violate the computer onto which it will be downloaded, the Server will perform tests on the program the check if the program never accesses memory locations that it is not supposed to. Such tests will ensure that the program never reads data that is private on the client computer, and does not write to forbidden locations and thereby self-replicate itself. Our tests will also detect if the program has memory leaks using state of the art techniques, to ensure that the program does not launch a denial of service attack on the client computer in this manner.

Our testing environment will run the program in a sandboxed environment to detect illegal memory accesses. One of the novelties of our testing approach will be to generate test inputs with large coverage to ensure that many of the program's computation paths are exercised. We will do this using a two-pronged approach. Random testing will be used to catch problems in commonly exercised paths. However, pure random testing will fail to exercise the corner cases with high probability. To account for that, we will perform symbolic execution that constructs a logical property characterizing the exercised paths. The SAT solvers will be used to generate inputs to exercise paths that have not been explored. Finally, to detect potential memory leaks we will run garbage collection routines periodically. Since all our tests will be performed before deployment (and not on the client computer), the overhead incurred due to the sandboxed environment, the symbolic execution, and memory leak detection, will not affect the performance of the program when it is used by clients.

Digital Signatures

Once the software is checked for security it is vital to ensure that only this version of the program will be downloaded and not an altered version. Therefore, a digital signature will be created for the software. The digital signature can be stored on the Sever and used along with a public key to verify that the software is indeed the original one that was first tested. The signature is much shorter than the program itself so checking it is much more efficient than comparing two copies of the entire software. It is secure based on some cryptographic assumption in the sense that if some bits of the software are changed than the signature will not be valid.

Component 2: User Experience

It is possible that even with the secure checks that will be made in Component 1, the software could be hacked after the download. In order to detect such problems, in Component 2 the Server will keep a score based on the responses of users of how well the software worked for them and whether they found any bugs or not. It is important to ensure that only responses of credible users will be used in the score calculation.

How do we Alert all Users if a Program has Complaints?

The approach we discuss here is based on a distributed agent-based model for network security as discussed above, which describes a distributed multi-agent system for real-time collection, monitoring, aggregation, and modeling of system and network operations, communications, internal and external access, coded execution functions, network and network resource conditions, as well as other assessable criteria. A Bayesian model is used to estimate likelihoods of various threat vectors. The model provides access to the reasoning behind its inferences. It may recommend or in some cases even implement responses to detected threats.

Component 3: Programmer Reputation

The Server will keep track of the identity of the programmer who created the software. Each time the software gets a good score from user's responses and from the security of the code the rating of the programmer will go up and this will be linked to all the different software that the programmer created.

There is an incentive for a programmer to create a user name that will be used for all the software that he creates. This way his reliability score is increased and his software will be more widely used. Even software that might be new that might not yet have a wide range of user responses in Component 2 will benefit from being linked to other software with high rating from the same programmer

If a programmer chooses to hide his identity from the software then there will be no score for Component 3 for this particular software. Therefore, there will be incentive for a good programmer to associate his username with the program.

If a programmer chooses to identify himself with the software, then there are two possible ways to verify the identity of the programmer. One is to identify the software with an identity such as an email address or another created user id. With this scheme a user may create different user identities for each software application that he creates. In this way the software will not benefit from being linked to other software created by the same programmer. Therefore there is an incentive for an honest programmer to use the same user id for all programs. The programmer usernames will be handed out with a password so different programmers will not be able to obtain the same user-id. Therefore, a malicious programmer will not be able to link his software with the credibility score of another programmer

A programmer that creates many programs will have further incentive to invest more time to create a Unique user id for which the Server requires paper identification such as a notarized copy of a driver's license. A programmer has the incentive to obtain such a Unique id because only one will be given to each programmer. Therefore, the programmer cannot create different programs that are not linked to this. If there is a virus in one of these programs then all the programs of this programmer will be given a bad score in either Component 1 or Component 2. Additionally, it is possible to trace the programs back to the programmer. Therefore, users looking for utmost credibility will look for software where the programmer identified himself with this Unique-id.

How do we Decide which Programs to Use?

A programmer may request to add his program onto the Server. Additionally, a user may request to add a program onto the site. The program does not need to be stored on the server. Only the signature can be stored on the server and as long as the program that can be downloaded from any site can pass the signature test it can be used.

Those skilled in the art will appreciate that this embodiment whereby a Server provides security and validity information about free online software that can be downloaded off the web can be implemented either as part of the SDI-SCAM system described above with respect to FIG. **1** or as a standalone scheme for improving computer security using the proposed credentialing scheme.

Whitelisting

This section highlights the value of leveraging the knowledge accumulated by the distributed agent of SDI-SCAM which particularly helps to overcome the limitations of whitelisting which in present day implementation is limited to a small fraction of the actual, but non-verifiable whitelist. Whitelisting particularly in the context of SDI-SCAM's probabilistic assessment of individuals, code, servers and user machines is a form of reputation system. Explicit whitelisting where a user is granted permissions to access certain servers or files or where they are considered safe individuals, machines, servers or code may, however, be used along with (or alternatively to) implicit whitelisting where such whitelisting functions are provided by SDI-SCAM based upon a probability distribution curve of safety or appropriateness of access (to typically specific programs, servers or machines) based upon matching or exceeding of a "safety threshold." The widely distributed agent based monitoring not only is capable of observing the large universe or network of networks but does it in a continuous and updated fashion. The non-whitelisted item would traditionally be "unknown" with the possibility of being unsafe or part of a black list in contrast to what is possible through the newly proposed paradigm in which what is not whitelisted has a very high probability of being unsafe.

Thus, in a global implementation of distributed agents, non-whitelisted machine and users, for example, could be for practical purposes effectively considered "unauthorized" in similar fashion to unauthorized "accessors" similarly for code or programmers in a closely monitored secure network. Compared to all known prior art counterpart systems, the use of a distributed sensor network to aggregate and provide means for data analysis and adaptive rules for modeling and predicting existing, emerging threats and threats which are yet to occur are particularly useful in early identification and prediction of the anticipated threats including anticipating severe threats (urgently warranting notifications and warnings), quantifying and/or characterizing servers, user machines, code, software and individuals regarding the present or predicted threat and revealing the likelihood of the threat (and/or potential severity or sophistication thereof). This includes the predicted safety or threat level, of a given user machine, server, program (including the user's own). Predictive threat level if high may also be revealed by black listing an item by implicit determination or if low whitelisting an item by matching or exceeding certain thresholds thus complementing explicit lists regarding the same with much more comprehensive data available through the distributed network of sensors.

In terms of one of its high level objectives SDI-SCAM strives for a comprehensive approach to higher efficiency computer security through the automation of these computer security processes. Thus, it is also useful to consider how it may be possible to improve not just the clarity in identifying and characterizing attacks and rogue activity but also how such information may be used to practically enable more

targeted and forceful defense measures and counter offense responses in such a fashion that because the actions taken are not based upon the assumption of the possibility of unsafe or malicious activity but rather a high degree of confidence if not certainty of such activity or intent. While many types of defense and counter offense measures exist and are well known in the art, there is nonetheless a fundamental tradeoff between implementing such defenses or counter offenses for the sake of improving overall security and/or creation of a deterrent and the overhead of implementation in combination with (most importantly) the additional time/effort and overhead navigating through those defenses and in the case of counter offenses the risk associated with launching the counter offenses against a non-malicious individual or machine. Because higher value servers that store the data tend to be the target of the more determined and skillful attacker, stronger defenses and counter offenses must be implemented for more valuable information and machines. Still, for the most part, these measures cannot be targeted because of the indefiniteness of the individual or program until it is too late and the attack has already occurred at which point typically even a counter attack is no longer possible.

As indicated, there are numerous types of defenses and counter offensive measures and it is clear that more targeted measures as a result of more definitive and confident identification and characterization of a true threat before the fact of actual attack will enable greater efficiency in protecting against or deterring threats and at lower cost and overhead. In the case of counter offensive measures there will be also much less collateral damage. Moreover, for this reason it will potentially enable much more common use of deterrent measures (typically counter offenses) be they human or automated by significantly revealing and characterizing the threat and leaving little question as to the intent to cause harm or steal information before the fact of the attack. Again, while there are numerous examples of how more common use of these measures under the widely distributed agent paradigm (ideally for these purposes an internet standard) there is listed a few examples of defensive and counter offensive measures below.

2. A family of security software which scan for software vulnerabilities and recommend security patches such as Protector Plus for Windows and Securia Personal Software Inspector. Clearly observational data generated across a large number of installations of any given software using distributed agents deployed to aggregate data, leverage relevant variables in a probabilistic model, test vulnerabilities in practice and develop probabilistic assessment of determined likely vulnerabilities would do a more effective and comprehensive job in the software vulnerability assessment, particularly identifying the more subtle or hidden vulnerabilities. Data mining and recommended testing protocols visa vie human in the loop approach is likely more effective than a fully automated analytic approach in this application.

2. Vulnerability exploitation prevention software, e.g., Malwarebytes Anti Exploit tool and ExploitShield—clearly by determining a more comprehensive profile of what existing vulnerabilities are present along with their characterization it is possible to better protect against their exploitation through a more targeted defense scheme. Again, the analysis of widely aggregated data and testing for each potential vulnerability is similar to that of item 1 above.

3. Performing direct counter offensive measures against an attacker—As indicated there are inherently significant risks of inadvertently attacking a machine which is not a

threat, for example, by forgetting to add that machine to the white list. Even if the risk is small users will see the potential downside is greater than the upside. They typically will use instead basic encryption to further protect the file. In the distributed agent environment, observational data will allow the agent to recognize certain machines (e.g. via their machine code) as a non-threat (whitelisted by implicit versus explicit means) through their patterns of usage, e.g., files which tend to be shared (or similarly accessed by) whitelisted machines particularly those with higher security access permission will indicate at least the strong possibility that the accessing machine is not a threat and should not be counter attacked, but instead further investigated.

Another measure which potentially alleviates the risk of "collateral damage" to (mistakenly) non-threatening machines is to not destroy data or applications on the presumed adversary but rather to simply disable the machine, for example, using a counter offensive program which invokes the disabling function via the secure operating system. While a destructive virus, e.g., which wipes clean the hard drive is a fairly strong potential deterrent, so would be a program which locks the operating system until/unless the suspect attacker can prove to the owner of the targeted machine that the suspected intrusion was a "false alarm," that no such presumably intentional unauthorized access had occurred. If the legitimacy of the "intruder" can be verified to the target machine owner, a remedy (or vaccine) may be provided to the previously presumed attacker's machine. For example, it may involve the owner possessing a vaccine needed to eliminate the virus or to unlock the operating system which the virus had locked, which is, metaphorically, an "antidote" to the harmful side effects of an aggressive drug treatment. Thus, while the counter attack would be as harmful to the would-be attacker as a traditional destructive virus, it would be potentially immediately reversible and innocuous to a mistaken attacker (perhaps a legitimate intended authorized party).

In short, what is proposed is a counteroffensive measure (which may be but not necessarily a Trojan) which upon infecting or otherwise accessing the attacker's (or unauthorized accessor's) machine invokes the operating system to shut down. The victim of the original presumed attack holds a vaccine to reverse the operating system shutdown which typically requires that the original attacker prove that the access was legitimate or authorized and/or for the victim to verify that no harm was done on his/her machine or network. The privilege to use the code or tool to disable the operating system may be revoked by the OS provider if abuse is detected (e.g. using it as a mainstream attack strategy instead of a counteroffensive one).

How the direct counter offensive measure is implemented typically requires entry into the attacker's machine. SDI SCAM if implemented on an ISP server local to the attacker or even in the form of a virus which infects the attacker's machine may be able to learn about the attacker, his preferred targets, groups or other contacts or hackers which he may be directly or more loosely affiliated with will allow not only a detailed "snapshot" of the attacker but also the otherwise hidden activities, targets, timing patterns, etc., of him and his affiliates (if he does not work alone). This information may indicate whether the attacker is interested in data, and what type (s), and/or software and what type(s) (e.g., is it for the construction of a botnet?). If certain software is of interest perhaps he can be lured and baited to upload a Trojan masquerading as a program of potential interest. The same can be performed for certain types of user and server machines and regarding certain associated content (different ways files can be executable to masquerade as harmless data files). Certainly, detailed activity logs uploaded to the SDI SCAM network could be used to help warn known targets (or types thereof), individuals, system administrators, other (local) SDI-SCAM agents, etc. of potential attack, the type of attack typical of that attacker, vulnerabilities which tend to be exploited by that attacker and/or as pertains to the particular type of target user. For example, certain types of honey pot traps unique to the attacker could be set up. Of course, this information is useful in implementing defensive and counter offensive measures and may be delivered to the SDI SCAM agent on each user's machine. For aiding potential SDI-SCAM or locally deployed counter offensives such detailed reporting may include software and potential vulnerabilities associated with the attacker himself. Even if the attacker causes significant harm to such entities as governments, financial institutions, credit card companies and corporate databases through theft of valuable data (as opposed to software code) it is possible to create executable code which mimics certain file types, or even modify certain content or file formats with latent executable functionality, e.g., Word or Excel. Such files can be completely dormant (non-executable) and thus innocuous if within a "friendly environment" e.g., an authorized machine code of the accessor's machine is recognized or it may be detected as a potential threat or alternatively eliminated by the local virus scanner which may be tied into the SDI-SCAM system which in this case is fully aware of the fact of the file being used as "bait" for attackers. SDI-SCAM may have planted it as part of a targeted and timed counter offensive thus keeping accurate record of adverse threats and friendly threats as part of the overall and up to date status of existing and likely behavior of threats as well as the success statistics for certain defensive and counter offensive measures in view of certain types of sophisticated attacks and attackers.

Therefore Trojans (such as the above mentioned) as well as other approaches can be automatically recommended and implemented to targeted rogue machines in order to launch a counter attack to an intrusion and unauthorized access to data or software. Such files may be accessed from a honey pot trap or such Trojans may (again via SDI SCAM administration) be strategically selected and positioned within the network to maximize the likelihood of a likely attacker to upload the Trojan containing the SDI SCAM agent (technically-speaking, acting as a virus). For the counter-offensive variation, SDI-SCAM (or its human analyst counterpart) will tend to select or construct those files as Trojans within a given target machine which it deems most likely to be accessed (of interest) by the attacker given the total data profile collected about the attacker's elicit data consumption patterns and relevance feedback from trial and error implementations which will indicate the preferred files based upon past attack or consumption patterns, the files and types of files snarfed by similar attackers or which perform similar types of attacks.

In summary, using SDI-SCAM's distributed adaptive learning model not only defensive (as discussed earlier) but also counteroffensive measures may be recommended and targeted in response to a given threat or attack which is provided by the system based upon relevance feedback including trial and error from previously delivered responses to previous threats and attacks. Such relevance feedback may update the system's adaptive rule base and utilize automated and human in the loop adaptive rules via data mining.

With regards to the state of the technical art of creating Trojans which appear to be innocent data files, currently, there is a significant gap between files and executable files. To do harm a file has to 1) be executable and 2) be executed. The latter requires that someone click on it or otherwise run it. This is very unlikely from a hacker who has snarfed the file from a target machine since he has no real incentive to run a captured executable?

There are only a few types of not apparently executable files can be executed—mostly Word and Excel files which are running VBA. It would be tricky to get a VBA macro that was able to figure out it was on an adversary's machine which is black listed or non-white listed (where in a global SDI-SCAM implementation is effectively black-listed) and then attack that machine successfully. SDI-SCAM's distributed knowledge may help here. Also, possible modifications to the file format to automatically invoke an execution, determine foreign, unauthorized machine environment, e.g. mac code and other clues matched against a list of authorized machines or white list (e.g., stored in upstream database). If a typical destructive virus was used, the maximum likely penalty would be the black hat would have to rebuild his box, not a huge deterrent factor. Unless the penalty was the virus' disabling the OS until/unless the presumed attacker could prove to the data owner his in fact authorization and/or non-malicious intent. Once word got out about it the tactic would become less effective until/unless adoption became fairly prevalent. Of course, designing the Trojan to look like an innocuous (non-executable) file format is another approach. The problem of its likely hitting innocent bystanders, i.e. laptops not preauthorized for the file but which should have been (e.g. destroying a vice president's laptop because he forgot to have a file authorized for his box) can be mitigated (again) by the use of SDI-SCAM to determine the likelihood of machines to be authorized for access to certain files (even if not explicitly) authorized. The default approach is to only encrypt the files instead of initiating counteroffensive measures. This of course, does not provide a deterrent and encryption keys can be stolen e.g. through break-ins and compromised digital certificates. Nonetheless, one of the present SDI-SCAM objectives is adapting the defensive and counteroffensive strategy by better selecting the response and making the response more targeted, confident and appropriate to the target entity, its actions and intervening circumstances/conditions in view of associated probabilities of all of the above.

For example, depending upon the value of the data (to the owner and attacker), the present circumstances, etc., SDI-SCAM can utilize its probabilistic knowledge as well as data as to the foreign environment (e.g. non-white listed) to determine the likelihood of a compromise of the encryption key thus possibly warranting the executable code associated with the file to prevent decryption e.g. by self-destruction of the file or for example, double encryption whereby the code holds the other second key which is obfuscated within other code or where the code (agent) sends it on a mix path (to be decrypted at a different server and returned) if and only if the environment of the initial accessor's (key holder's) machine is whitelisted or otherwise authorized.

4) As described earlier, the probabilistic model for SDI-SCAM's distributed agents does a much better more comprehensive and (most remarkably) a faster job at detecting patterns in software, behavior, sequences, etc., than could be achieved by other state of the art means like virus scanning software. Nonetheless, the fact remains that a lot of rogue software is new, evolving, difficult to detect (e.g., dormant) or maybe "normal" software with hidden vulnerabilities readily exploitable by hackers and/or malware. Probabilistic modeling of likely software, servers and machines that may be unsafe should include the communication patterns indicating the possibility of infection by unintentional access to rogue machines, servers, software or individuals including communication with those software, servers and machines which may have come in contact with others which are suspicious. Such communication patterns (particularly changing ones) may also provide important clues of after the fact infection once a machine has been infected and taken control of e.g., from a hacker or a botnet. The other point to be made in this regard is that code (and by extension is associated machines) are particularly vulnerable to attack if the code itself is not secure. Thus, in determining probability of infection or intrusion, it is useful to apply SDI-SCAM's probabilistic modeling to the assessment of detection, likelihood and characterization of vulnerabilities of each piece of software on the network which, in turn, feeds into the broader general purpose probability analysis of intrusion and infection of software and machines across the network as herein explained.

The persistent distributed agent monitoring is particularly effective in monitoring rogue software at work in the network (or in one of SDI-SCAM's quarantined environments) and hackers being observed (in practice or in one of the system's honey pot traps) is revealing (thus can be data mined autonomously or via human in the loop) to observe and predict potential triggers (e.g. conditions, environments, data, potential targets) which certain types of suspected or identified rogue code or individuals will likely respond to so as to thus elicit their true nefarious natures or to test other triggers to learn more about them. Thus, the adaptive learning model can be learning by not just observing but also by learning triggers that certain types/characterized known "bad actors" tend respond to in a revelatory way thus providing an optimal testing strategy for triggering identifying information about it (is it unsafe or not) and secondly, what are its malicious characteristics or ways in which it can do harm. Particularly when dealing with minimal information a decision tree is a good type of decision-making tool to assess the best tests/sequence thereof for the actor based upon its own limited data profile in view of the overall probabilistic model of the system. This is because a decision tree is very efficient at spanning the space of probabilistically most likely heuristics (via associated tests) most quickly and efficiently in order from highest to lowest relevancy.

In addition to its role in providing certain key knowledge to the overall probabilistic assessment of probability and characterization of infection, intrusion and rogueness of programs, machines and individuals across the network, the application of SDI-SCAM's distributed intelligence providing a "telescopic view" of symptomatic software vulnerabilities is of course also key to ascertaining the identification and characterization of vulnerabilities (particularly the less obvious ones). It may readily tie into thus making more adaptive and comprehensive in a total data volume and data analytic sense existing rule sets for vulnerability detection software and vulnerability anti-exploitation software as well as determining the most effective solutions to recommend to remedy the identified problem, temporarily and more permanently (e.g. patches or redesign in further iterations). SDI-SCAM's distributed sensor network is particularly effective at aggregating and analyzing data across a plethora of installations of a software program, monitoring potentially a variety of heuristics indicative of potential vulnerabilities. Human in the loop or autonomous data analytic insights are thus provided into what vulnerabilities exist and what updates, patches etc. to use in future iterations of the software. Adaptive learning based upon relevance feedback from trial and error may be used to make recommendations thus providing refinements and evolve an adaptive model in order to make recommendations for example for more effective shorter-term solutions such as recommending patches and anti-exploitation solutions which are more specifically targeted to the specific vulnerabilities identified and characterized. In short, SDI-SCAM should do well in detecting, characterizing etc. vulnerabilities particularly so for applying its distributed sensor intelligence to recommending and delivering potentially a host preemptive defensive measures once it has identified and characterized existing and potential vulnerabilities including the possibility of recommending further tests to verify and/or further characterize them.

Tell-tale signs of failure of a software program, a machine or even entire computer network which is insecure could be a big data problem using perhaps non-linear kernel regression methods. Sequential progression/escalation of conditions which presage slowdown or failure are detectable even when such conditions are manifested via combinatorial heuristics. SDI-SCAM's probabilistic comprehensive assessment of vulnerabilities and comprehensive threats can feed into an alternative predictive model developed for predicting failures. In other applications of this concept, a large variety of sensors are used across a plethora of the mechanical systems (particularly engines) of commercial aircraft, fed into a database for "big data" data mining and improved prevention and diagnosis of both impending problems (on an individual basis) and general areas of vulnerability (on a general system engineering level) are all possible. In any event, such a system-level application of SDI-SCAM can make prescient predictions of "trigger" conditions or actions to avoid getting into a "yellow" or caution alert state and those to avoid to prevent getting into a red state (failure or impending failure) once in a yellow state. Or conversely, remediatory actions given any set of current conditions (which are evolving and adaptively learned by trial and error). Of course, the determination of the very fact that conditions characterizing yellow state potentially presaging red state are also part of the big data analysis. Determination/detection and best response (remedial actions) for each associated (probabilistic) alert state and associated set of conditions all seem feasible. Time sequence analytical approaches (e.g. non-linear kernel regression methods) are particularly useful in modeling, identifying responding to (or remediating) temporally evolving system states.

Previous discussions have focused principally upon determining/characterizing general system vulnerabilities (of overall value in software system (re) design and upgrading) but could be useful with regards to individual time-sensitive detection, response/remediation, problem or failure likelihood assessment as well as overall preventative operations strategies for a given software system. The same strategy of big data collection involving a variety (perhaps many) sensors across a large number of objects may be used in determining early warning detection/characterization profiles elucidating presence of conditions associated with (or presaging) potential failures for other types of complex systems like the grid (e.g. blackouts) where SDI-SCAM network of agents monitor for intrusions and infections as well as a host of other variables which are well known in the field of the particular art.

Those skilled in the art will also appreciate that the invention may be applied to other applications and may be modified without departing from the scope of the invention. Accordingly, the scope of the invention is not intended to be limited to the exemplary embodiments described above, but only by the appended claims.

What is claimed is:

1. A system that detects the state of a computer network, comprising:

a plurality of distributed agents disposed in said computer network, each said distributed agent including a microprocessor adapted to:

passively collect, monitor, and aggregate data representative of activities of respective nodes within said computer network,

analyze collected data to develop activity models representative of activities of said computer network in a normal state and activities of said computer network in an abnormal state as a result of intrusions, infections, scams, code emulating code or humans, and/or other suspicious activities in said computer network, and

generate counter-offensive measures where unauthorized access to a program or file containing executable code results in the program or file disabling an operating system with all associated applications of a computer in the computer network until/unless the presumed attacker is able to prove to the machine owner/victim that the presumed attacker had been authorized to access the target data or machine provoking the said counter offensive measure; and

a server that provides a security and validity score for free software available for download, the validity score comprising three components including a first component computed based on security of the free software itself, a second component computed based on experiences users have with the free software, and a third component based on a reputation of a programmer that created the free software.

2. A system as in claim **1**, further comprising a distributed sensor network that aggregates and analyzes data to develop a probabilistic likelihood of a threat to safe code, machines, servers, or individuals, wherein said counter-offensive generating means generates counter-offensive measures that are targeted to a given threat or attack based upon historical feedback from successes and failures of previous counter measures used in response to similar attacks and threats.

3. A system as in claim **2**, wherein said historical feedback updates an adaptive learning or adaptive rule base.

4. A system as in claim **2**, wherein said distributed sensor network aggregates and analyzes data pertaining to software within the network in order to enable detection and characterization of vulnerabilities and/or provide recommendations for remedial repair or revision.

5. A system as in claim **1**, wherein said counter-offensive generating means creates a bogus target for invoking attack on said bogus target for purposes of achieving early detection of a system infection.

6. A system as in claim **1**, wherein each distributed agent is connected to each other distributed agent through a fully isolated computer network that operates independently from the computer network on which a protected computer resides.

7. A system as in claim **1**, wherein when the analyzing means detects suspicious activities, a distributed agent opens a "honey pot" trap in a virtual space that simulates an environment of the protected computer.

8. A system as in claim **1**, wherein each distributed agent monitors processes for behavior consistent with viral infection and flags processes having said behavior as a potential threat, said monitoring including determining whether a process opens and modifies a wide range of heterogeneous files, whether a process accesses a mail system's address folder, whether a process aggressively propagates copies of itself, whether a process engages in recursively redundant actions whose objective is designed to achieve no useful purposes, whether a process aggressively or repetitively generates or obtains data files in order to propagate inordinately voluminous and/or large files resulting in bursts of traffic, whether a process performs similar recursively redundant actions resulting in consumption and overloading of valuable processing capacity, whether a process modifies or mutates its own code and/or behavior, and/or whether a process opens unexpected communication ports with outside entities will be flagged as a potential threat.

9. A system as in claim **1**, wherein each distributed agent determines a probability and a degree of ill motive of individuals of most likely suspicion and monitors activities of said individuals in the computer network.

10. A system as in claim **1**, wherein the server performs tests on the free software to check if the free program ever accesses memory locations that it is not supposed to access and to detect if the free software has memory leaks that would enable the free software to launch a denial of service attack.

11. A system as in claim **1**, wherein each distributed agent monitors time and circumstances of any changes to the computer network, cross-checks the time and circumstances of any changes against average traffic patterns for the computer network, and flags changes at unusual times or under unusual circumstances as suspicious activities.

12. A method of detecting the state of a computer network, comprising:

providing a plurality of distributed agents disposed in said computer network to passively collect, monitor, and aggregate data representative of activities of respective nodes within said computer network;

analyzing said data using a microprocessor to develop activity models based on collected data and representative of activities of said network in a normal state and activities of said computer network in an abnormal state as a result of intrusions, infections, scams, code emulating code or humans, and/or other suspicious activities in said computer network, said data analysis including performing a pattern analysis on the collected data to identify patterns in the collected data representative of suspicious activities;

generating counter-offensive measures where unauthorized access to a program or file containing executable code results in the program or file disabling an operating system with all associated applications of a computer in the computer network until/unless the presumed attacker is able to prove to the machine owner/victim that the presumed attacker had been authorized to access the target data or machine provoking the said counter offensive measure; and

a server providing a security and validity score for free software available for download, the validity score comprising three components including a first component computed based on security of the free software itself, a second component computed based on experiences users have with the free software, and a third component based on a reputation of a programmer that created the free software.

13. A method as in claim **12**, further comprising aggregating and analyzing data to develop a probabilistic likelihood of a threat to safe code, machines, servers, or individuals, and generating counter-offensive measures that are targeted to a given threat or attack based upon historical feedback from successes and failures of previous counter measures used in response to similar attacks and threats.

14. A method as in claim **13**, further comprising updating an adaptive learning or adaptive rule base with said historical feedback.

15. A method as in claim **13**, further comprising aggregating and analyzing data pertaining to software within the network in order to enable detection and characterization of vulnerabilities and/or provide recommendations for remedial repair or revision.

16. A method as in claim **12**, further comprising creating a bogus target for invoking attack on said bogus target for purposes of achieving early detection of a system infection.

17. A method as in claim **12**, wherein each distributed agent is connected to each other distributed agent through a fully isolated computer network that operates independently from the computer network on which a protected computer resides.

18. A method as in claim **12**, wherein when the analyzing means detects suspicious activities, a distributed agent opens a "honey pot" trap in a virtual space that simulates an environment of the protected computer.

19. A method as in claim **12**, further including each distributed agent monitoring processes for behavior consistent with viral infection and flagging processes having said behavior as a potential threat, said monitoring including determining whether a process opens and modifies a wide range of heterogeneous files, whether a process accesses a mail system's address folder, whether a process aggressively propagates copies of itself, whether a process engages in recursively redundant actions whose objective is designed to achieve no useful purposes, whether a process aggressively or repetitively generates or obtains data files in order to propagate inordinately voluminous and/or large files resulting in bursts of traffic, whether a process performs similar recursively redundant actions resulting in consumption and overloading of valuable processing capacity, whether a process modifies or mutates its own code and/or behavior, and/or whether a process opens unexpected communication ports with outside entities will be flagged as a potential threat.

20. A method as in claim **12**, further including each distributed agent determining a probability and a degree of ill motive of individuals of most likely suspicion and monitoring activities of said individuals in the computer network.

21. A method as in claim **12**, further including the server performing tests on the free software to check if the free program ever accesses memory locations that it is not supposed to access and to detect if the free software has memory leaks that would enable the free software to launch a denial of service attack.

22. A method as in claim **12**, further including each distributed agent monitoring time and circumstances of any changes to the computer network, cross-checking the time and circumstances of any changes against average traffic patterns for the computer network, and flagging changes at unusual times or under unusual circumstances as suspicious activities.

* * * * *

Exhibit D

to

Complaint
for Patent Infringement

The ’974 Patent

US011171974B2

(12) **United States Patent**
Gertner et al.

(10) **Patent No.: US 11,171,974 B2**
(45) **Date of Patent: *Nov. 9, 2021**

(54) **DISTRIBUTED AGENT BASED MODEL FOR SECURITY MONITORING AND RESPONSE**

(71) Applicant: **Fred Herz Patents, LLC**, Milton, WV (US)

(72) Inventors: **Yael Gertner**, Champaign, IL (US); **Frederick S. M. Herz**, Milton, WV (US); **Walter Paul Labys**, Fairfax, VA (US)

(73) Assignee: **Inventship LLC**, Warrington, PA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **15/357,399**

(22) Filed: **Nov. 21, 2016**

(65) **Prior Publication Data**

US 2017/0078317 A1 Mar. 16, 2017

Related U.S. Application Data

(63) Continuation of application No. 14/043,567, filed on Oct. 1, 2013, now Pat. No. 9,503,470, which is a (Continued)

(51) **Int. Cl.**
H04L 29/06 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 63/1425*** (2013.01); ***H04L 63/145*** (2013.01); ***H04L 63/1416*** (2013.01); ***H04L 63/1441*** (2013.01)

(58) **Field of Classification Search**
CPC H04L 63/145; H04L 63/1425
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,754,938	A	5/1998	Herz et al.
5,754,939	A	5/1998	Herz et al.

(Continued)

OTHER PUBLICATIONS

U.S. Appl. No. 61/702,453, filed Sep. 12, Akshay Vashist.*

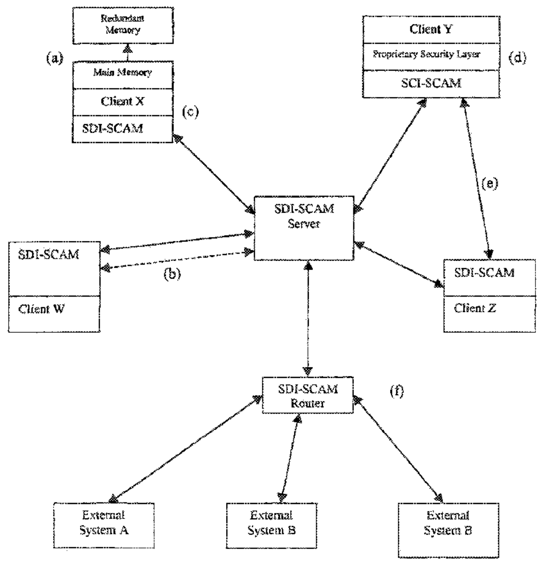
(Continued)

Primary Examiner — Jeffrey C Pwu
Assistant Examiner — Thong P Truong
(74) *Attorney, Agent, or Firm* — Culhane Meadows, PLLC

(57) **ABSTRACT**

An architecture is provided for a widely distributed security system (SDI-SCAM) that protects computers at individual client locations, but which constantly pools and analyzes information gathered from machines across a network in order to quickly detect patterns consistent with intrusion or attack, singular or coordinated. When a novel method of attack has been detected, the system distributes warnings and potential countermeasures to each individual machine on the network. Such a warning may potentially include a probability distribution of the likelihood of an intrusion or attack as well as the relative probabilistic likelihood that such potential intrusion possesses certain characteristics or typologies or even strategic objectives in order to best recommend and/or distribute to each machine the most befitting countermeasure(s) given all presently known particular data and associated predicted probabilistic information regarding the prospective intrusion or attack. If any systems are adversely affected, methods for repairing the damage are shared and redistributed throughout the network.

16 Claims, 1 Drawing Sheet



Related U.S. Application Data

continuation-in-part of application No. 10/746,825, filed on Dec. 24, 2003, now Pat. No. 8,327,442, which is a continuation-in-part of application No. 10/693,149, filed on Oct. 23, 2003, now Pat. No. 8,046,835.

(60) Provisional application No. 61/708,304, filed on Oct. 1, 2012, provisional application No. 60/436,363, filed on Dec. 24, 2002.

(56) **References Cited**

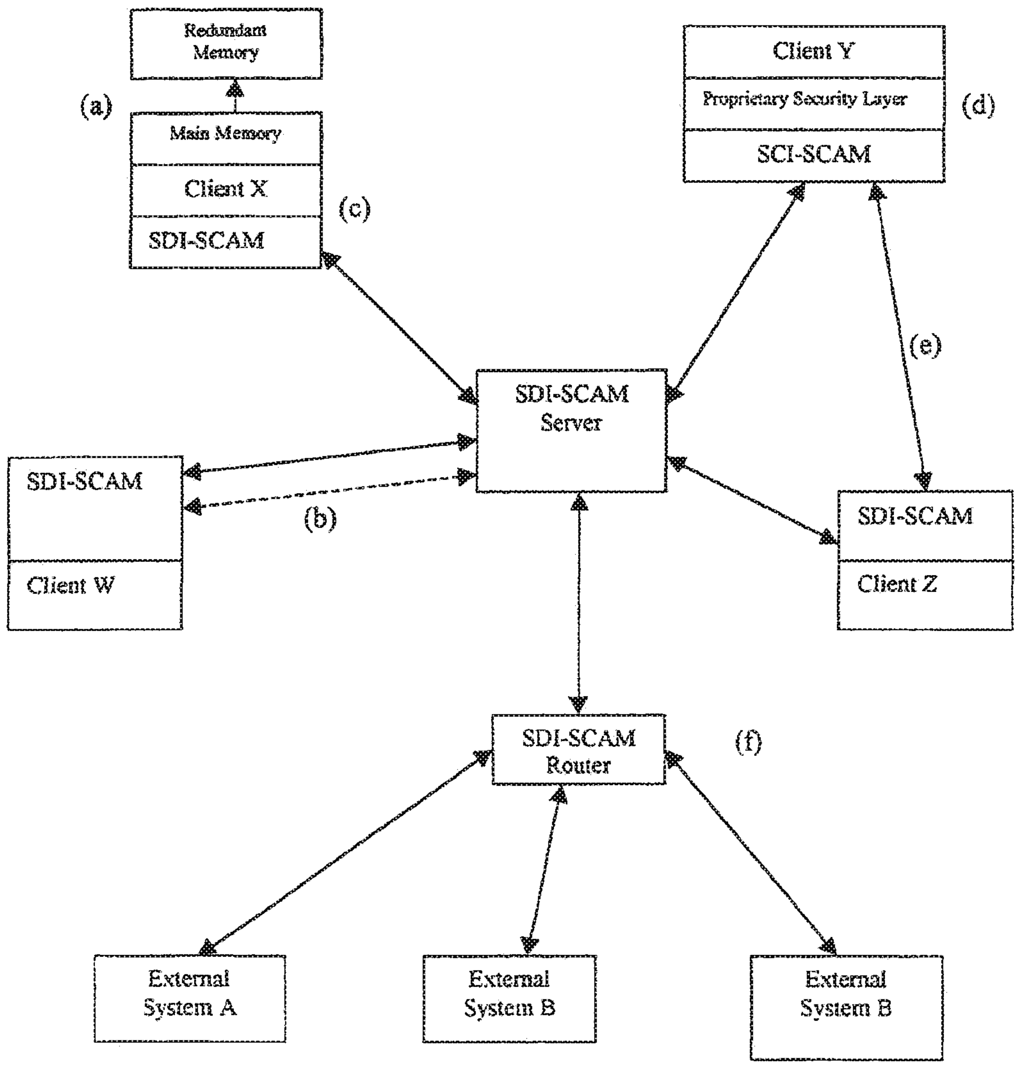
U.S. PATENT DOCUMENTS

6,088,804	A	7/2000	Hill et al.	
6,122,743	A	9/2000	Shaffer et al.	
6,158,010	A	12/2000	Moriconi et al.	
6,301,668	B1	10/2001	Gleichauf et al.	
6,345,299	B2	2/2002	Segal	
6,347,338	B1	2/2002	Segal	
6,353,385	B1	3/2002	Molini	
6,405,318	B1	6/2002	Rowland	
6,530,024	B1	3/2003	Proctor	
6,597,777	B1	7/2003	Ho	
6,663,000	B1	12/2003	Muttik et al.	
6,711,615	B2	3/2004	Porras et al.	
6,725,377	B1	4/2004	Kouznetsov	
6,742,128	B1 *	5/2004	Joiner	H04L 63/1408 726/25
6,769,066	B1	7/2004	Botros et al.	
6,772,349	B1	8/2004	Martin	
6,952,779	B1	10/2005	Cohen et al.	
6,996,843	B1	2/2006	Moran	
7,013,330	B1	3/2006	Tarbotton et al.	
7,058,968	B2	6/2006	Rowland et al.	
7,181,769	B1	2/2007	Kianini et al.	
7,185,368	B2	2/2007	Copeland, III	
7,222,366	B2	5/2007	Bruton, III	
7,228,564	B2	6/2007	Raikar et al.	
7,350,069	B2	3/2008	Herz et al.	
7,594,260	B2	9/2009	Porras et al.	
7,630,986	B1	12/2009	Herz et al.	
8,046,835	B2	10/2011	Herz	
8,327,442	B2	12/2012	Herz et al.	
8,490,197	B2	7/2013	Herz	
8,490,497	B2	7/2013	Herz	
8,925,095	B2	12/2014	Herz et al.	
9,438,614	B2	9/2016	Herz	
9,503,470	B2	11/2016	Gertner et al.	
2002/0059078	A1	5/2002	Valdes et al.	
2002/0066034	A1	5/2002	Schlossberg et al.	
2002/0067832	A1 *	6/2002	Jablon	H04L 9/0844 380/277
2003/0004688	A1 *	1/2003	Gupta	G06F 21/55 702/188
2003/0051163	A1	3/2003	Bidaud	
2003/0101260	A1	5/2003	Dacier et al.	
2003/0115322	A1	6/2003	Moriconi et al.	
2003/0145225	A1	7/2003	Bruton, III et al.	
2004/0123142	A1	6/2004	Dubal et al.	
2004/0153666	A1	8/2004	Sobel et al.	
2004/0165588	A1	8/2004	Pandya	
2004/0215972	A1	10/2004	Sung et al.	
2004/0250060	A1	12/2004	Diep	
2005/0257247	A1	11/2005	Moriconi et al.	
2006/0069749	A1	3/2006	Herz et al.	
2006/0259970	A1 *	11/2006	Sheymov	G06F 21/55 726/23
2008/0071578	A1	3/2008	Herz et al.	
2011/0078797	A1 *	3/2011	Beachem	G06F 21/53 726/25
2012/0131674	A1	5/2012	Wittenschlaeger	
2014/0082730	A1 *	3/2014	Vashist	H04L 63/1416 726/23

OTHER PUBLICATIONS

"U.S. Appl. No. 10/746,825, Advisory Action dated Mar. 11, 2008", 3 pgs.
"U.S. Appl. No. 10/746,825, Final Office Action dated Mar. 1, 2010", 26 pgs.
"U.S. Appl. No. 10/746,825, Final Office Action dated Apr. 24, 2009", 20 pgs.
"U.S. Appl. No. 10/746,825, Final Office Action dated Oct. 10, 2007", 21 pgs.
"U.S. Appl. No. 10/746,825, Non Final Office Action dated Jan. 24, 2011", 29 pgs.
"U.S. Appl. No. 10/746,825, Non Final Office Action dated Jul. 16, 2008", 20 pgs.
"U.S. Appl. No. 10/746,825, Non Final Office Action dated Aug. 28, 2009", 24 pgs.
"U.S. Appl. No. 10/746,825, Non Final Office Action dated Sep. 15, 2011", 31 pgs.
"U.S. Appl. No. 10/746,825, Non Final Office Action dated Dec. 7, 2006", 11 pgs.
"U.S. Appl. No. 10/746,825, Notice of Allowance dated Aug. 1, 2012", 30 pgs.
"U.S. Appl. No. 10/746,825, Response filed Jan. 16, 2009 to Non Final Office Action dated Jul. 16, 2008", 13 pgs.
"U.S. Appl. No. 10/746,825, Response filed Feb. 11, 2008 to Final Office Action dated Oct. 10, 2007", 43 pgs.
"U.S. Appl. No. 10/746,825, Response filed Mar. 15, 2012 to Non Final Office Action dated Sep. 15, 2011", 18 pgs.
"U.S. Appl. No. 10/746,825, Response filed Arp. 10, 2008 to Advisory Action dated Mar. 11, 2008", 15 pgs.
"U.S. Appl. No. 10/746,825, Response filed Jun. 7, 2007 to Non Final Office Action dated Dec. 7, 2006", 32 pgs.
"U.S. Appl. No. 10/746,825, Response filed Jun. 24, 2011 to Non Final Office Action dated Jan. 24, 2011", 19 pgs.
"U.S. Appl. No. 10/746,825, Response filed Jul. 24, 2009 to Final Office Action dated Apr. 24, 2009", 13 pgs.
"U.S. Appl. No. 10/746,825, Response filed Aug. 2, 2010 to Final Office Action dated Mar. 1, 2010", 15 pgs.
"U.S. Appl. No. 10/746,825, Response filed Nov. 30, 2009 to Non Final Office Action dated Aug. 28, 2009", 15 pgs.
"U.S. Appl. No. 14/043,567, Non Final Office Action dated Oct. 30, 2015", 16 pgs.
"U.S. Appl. No. 14/043,567, Notice of Allowance dated Jul. 18, 2016", 18 pgs.
"U.S. Appl. No. 14/043,567, Preliminary Amendment filed Sep. 1, 2015", 8 pgs.
"U.S. Appl. No. 14/043,567, Response filed Mar. 30, 2016 to Non Final Office Action dated Oct. 30, 2015", 11 pgs.
Anderson, et al., "Next Generation Intrusion Detection Expert System (NIDES) A Summary", SRI-CSL-97-07, (May 1995), 47 pgs.
Denning, D E, "An Intrusion-Detection Model", IEEE Transactions on Software Engineering, vol. SE-13, No. 2, (Feb. 1987), 222-232.

* cited by examiner



Redundant Memory
(a)
Main Memory
Client X
SDI-SCAM
(c)
Client Y
Proprietary Security Layer
(d)
SCI-SCAM
(e)
SDI-SCAM Server
SDI-SCAM
Client W
(b)
SDI-SCAM
Client Z
SDI-SCAM Router
(f)
External System A
External System B
External System B

DISTRIBUTED AGENT BASED MODEL FOR SECURITY MONITORING AND RESPONSE

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is a continuation of U.S. patent application Ser. No. 14/043,567 filed Oct. 1, 2013, which claims benefit to U.S. Provisional Application No. 61/708,304 filed Oct. 1, 2012, and which is a Continuation-in-Part of application Ser. No. 10/746,825, filed Dec. 24, 2003, now U.S. Pat. No. 8,327,442, which is, in turn, a Continuation-in-Part of application Ser. No. 10/693,149, filed Oct. 23, 2003, now U.S. Pat. No. 8,046,835, that further claims benefit of Provisional Patent Application 60/436,363, filed Dec. 24, 2002. These patent applications are incorporated herein by reference in their entireties.

BACKGROUND OF THE INVENTION

(1) Field of the Invention

The invention related to the field of security systems for computer networks.

(2) Description of Related Art

The United States and the rest of the world present a target rich environment for a variety of cyber threats. Rogue hackers (e.g. Anonymous) have created significant disruptions. Criminal organizations have employed botnets and other strategies to engage in massive wealth transfers. And state-affiliated actors have penetrated a number of US entities, including the US Chamber of Commerce, Nortel, and others.

Particularly troubling is the fact that many of the activities have gone undetected for significant periods of time, suggesting that what we have seen is only the tip of the iceberg. Further, while much of this activity has been aimed at achieving political, financial, or diplomatic advantage, there is the troubling possibility of a coordinated attack taking out critical infrastructure in military, industrial, power generation, financial and other critical centers. If a previously prepared attack were coordinated with a significant conventional threat there could be global ramifications.

Per NSA Director Keith Alexander, perhaps a trillion dollars a year is being spent on cyber defense. This has not, however, bought a trillion dollars' worth of confidence in our cyber defenses.

It has proved difficult to bring threatened institutions up to currently recognized levels of best practice, i.e. use of secure passwords, single sign-on, least privilege, multiple firewalls, and so on. And it is far from clear that even current best practice is "best" enough.

Further, the large expenditures themselves represent a kind of failure. The current cyber-attacks represent a strongly asymmetric form of warfare. Individuals and small groups can create extraordinary DDOS attacks with only limited resources. The increasing complexity of software creates an exponential increase in potential points of attack. Increasing sophistication in hacks, e.g. self-assembling viruses, is providing new ways to exploit weakness. And the general sloppiness of the web—together with its use for more and more critical infrastructure—generates what one might euphemistically refer to as a "negative progress situation".

If attackers can spend small quantities of resource while generating large, expensive, slow, and relatively ineffectual responses, then even a successful response, if expensive enough, may mean a net strategic fail.

If we are to reverse this trend we will need approaches which are:

1. Adaptive
2. Autonomous
3. Automatic

They must be adaptive because the threats are, autonomous because the threats are high frequency and unceasing, and automatic because human response times are too great.

What is desired is something like an immune system for software, where even novel threats are recognized quickly and kick off a well-defined cascade of defensive and prophylactic measures, without conscious attention. Ideally the hyper-caffeinated hackers might spend weeks devising a new line of attack, only to see it flagged as abnormal and countered in milliseconds. The inventors have, ultimately, no objection to asymmetry, and would just like to see the sharp end of the asymmetry pointing in the other direction.

This is a non-trivial problem. Given the significance and difficulty of the problem, sound principles of portfolio management require exploration of a wide variety of approaches.

One approach to this problem is a distributed agent-based model for network security described in U.S. Pat. No. 8,046,835, and incorporated above by reference. As explained further below, this patent describes a distributed multi-agent system for real-time collection, monitoring, aggregation, and modeling of system and network operations, communications, internal and external access, coded execution functions, network and network resource conditions, as well as other assessable criteria. A Bayesian model is used to estimate likelihoods of various threat vectors. The model provides access to the reasoning behind its inferences. It may recommend or in some cases even implement responses to detected threats.

Since the time that the subject matter described in U.S. Pat. No. 8,046,835 was developed, cloud and other technologies have made tests of it significantly more feasible in the last few years. The basic idea is a large number of sensors, aggregators, and other agents monitor an at-risk system looking for anomalies. Bayesian analysis is used to estimate the probabilities that a particular pattern of activity is hostile. In the most sensitive cases, any variation from established baselines might be considered potentially hostile. Many of the agents could live in the secure cloud, keeping them from putting too much of a load on the defended system, letting their activity ramp up quickly when threat levels are higher, and keeping them themselves from being a target.

Since the individual agents are simple, once the overall architecture has been validated, tuning the system to respond to new threats and opportunities should be rapid, being often merely a matter of writing a small agent & telling the system to listen to it. This is analogous to the way "plug-in" modules are currently used to quickly augment the capabilities of browsers, word processors and the like. For example, if software certification authorities become generally available, certification agents could verify firmware and executables against published checksums, at random intervals or when an attack is suspected.

Calibration of the Bayesian model is of course key. Banks of reference systems (i.e. "honey pots")—some prepared clean, others prepared with known threats present—could be used to tune & validate the model. Use of cloud computing makes it economical to run thousands of tests simultaneously, ideally giving a relatively accurate way to judge under what conditions the model can be allowed to trigger an automated response and when verification by a human operator would be first required.

While the inventors are now focused on the cyber security problem, the original system was meant to be more general in application, with network security, financial, medical, and other applications. The agents can be used to:

1. Establish baselines
2. Identify variations
3. Generate smaller groups of agents to target specific threats (vaccines)
4. Initiate automated responses, as raising firewalls, switching to a spare machine not under attack, and so on
5. Serve as laboratories for developing counter-agents, verifying pre-deployment they will be effective against their intended target with a minimum of collateral damage.

Given that the development of such distributed agent-based models for security modeling and response is now feasible, the next step is develop some reference implementations, to test the ideas in practice. One approach is to work with existing vendors, i.e. Cogility or Blue Canopy, to see how this agent-based system can help automate their existing threat detection approaches. Another is to work with vanilla Linux systems, again with the goal of automating the synthesis of existing monitoring tools into an adaptive, autonomous, and automatic security monitoring and response system. The present invention addresses these needs in the art.

BRIEF SUMMARY OF THE INVENTION

An architecture is provided for a widely distributed security system (SDI-SCAM) that protects computers at individual client locations, but which constantly pools and analyzes information gathered from machines across a network in order to quickly detect patterns consistent with intrusion or attack, singular or coordinated. When a novel method of attack has been detected, the system distributes warnings and potential countermeasures to each individual machine on the network. In a preferred implementation, such a warning may potentially consist of a probability distribution of the likelihood of an intrusion or attack as well as the relative probabilistic likelihood that such potential intrusion possesses certain characteristics or typologies or even strategic objectives in order to best recommend and/or distribute to each machine the most befitting counter-measure(s) given all presently known particular data and associated predicted probabilistic information regarding the prospective intrusion or attack. If any systems are adversely affected, methods for repairing the damage are shared and redistributed throughout the network. The net impact of SDI-SCAM is that every machine on a network can benefit from security experience gained at any other point on the network. A high and uniform level of security is therefore assured to all systems attached to the network, and this security is updated in real-time.

In exemplary embodiments, systems and methods are provided for detecting the state of a computer network by providing a plurality of distributed agents disposed in the computer network to passively collect, monitor, and aggregate data representative of activities of respective nodes within said computer network. The aggregated data is analyzed to develop activity models based on collected data and representative of activities of the network in a normal state and activities of the computer network in an abnormal state as a result of intrusions, infections, scams and/or other suspicious activities in the computer network. The data analysis includes performing a pattern analysis on the collected data to identify patterns in the collected data representative of suspicious activities. Counter-offensive measures are generated where unauthorized access to a program or file containing executable code results in the program or file disabling an operating system with all associated applications of a computer in the computer network until/unless the presumed attacker is able to prove to the machine owner/victim that the presumed attacker had been authorized to access the target data or machine provoking the said counter offensive measure.

The systems and methods of the exemplary embodiments also may aggregate and analyze data to develop a probabilistic likelihood of a threat to safe code, machines, servers, or individuals, and generate counter-offensive measures that are targeted to a given threat or attack based upon historical feedback from successes and failures of previous counter measures used in response to similar attacks and threats. An adaptive learning or adaptive rule base may be updated with the historical feedback. The systems and methods also may aggregate and analyze data pertaining to software within the network in order to enable detection and characterization of vulnerabilities and/or provide recommendations for remedial repair or revision. In addition, the systems and methods may generate a bogus target for invoking attack on the bogus target for purposes of achieving early detection of a system infection.

These and other characteristic features of the invention will become apparent to those skilled in the art from the following description.

BRIEF DESCRIPTION OF THE DRAWINGS

The FIGURE demonstrates some of the architectural features discussed, including (a) redundant memory within a given machine, (b) redundant connections between clients and servers, (c) SDI-SCAM installed as a primary security system, (d) SDI-SCAM piggybacking on an existing security system, (e) direct client-to-client agent communications, (f) on a router.

DETAILED DESCRIPTION OF THE INVENTION

The basic architectural approach for SDI-SCAM as described in U.S. Pat. No. 8,046,835 is that each node of a computer network is loaded with an agent capable both of ensuring security at the locality of the machine on which it is installed, and of communicating with other SDI-SCAM agents across the network. Because agent configurations are highly flexible, SDI-SCAM implementations can vary widely, running the spectrum from fully centralized (in which SDI-SCAM agents on client machines communicate uniquely with a centralized server dedicated to processing security-related information) to fully distributed (in which each client agent is given the ability to process security information locally, and information is shared on a peer-to-peer basis).

Basic Network Elements of SDI-SCAM

The preexisting elements of this network security system are the machines themselves. It is assumed that these systems, which act as the nodes of a network, consist of heterogeneous pieces of hardware running different sorts of operating systems. It may well be the case that various security layers will already be in place.

Additional Hardware

In preparation for the installation of SDI-SCAM across a network, it will often be desirable to upgrade existing machines with redundant hardware.

In a preferred embodiment, preexisting systems will be supplemented with redundant memory systems that persistently mirror the contents of the primary memory banks. When a computer's primary memory is corrupted (as can happen during a viral attack), it can be completed, cleared and reset with a pre-corruption image from the backup.

A further redundancy can be built into the network connections that link the local nodes to SDI-SCAM servers. For example, a computer that normally operates through land-based optic lines may be given an additional wireless connection through a satellite system.

An expensive, but preferred, architecture is to connect each SDI-SCAM agent through a fully isolated network that operates independently from the network on which the protected system resides. Thus, the SDI-SCAM agent will remain in contact with the security network even when the system it is supporting is under a sustained or unusually intense attack.

SDI-SCAM Agents

An agent is an entity that can be loaded onto any node(s) of a network, and which in this case is assigned responsibilities related to system security. Note that the construction of a given agent can vary widely, as it can be implemented through software, through hardware, through human interaction, or some combination thereof. In a preferred embodiment of SDI-SCAM, every machine linked into the system is loaded with an SDI-SCAM agent. Agent responsibilities include the following:

1) The collection of traffic data—among other things, each agent observes the packets being routed through its local system, observes every file transmission, monitors every user action, and logs every request for access.

2) The ability to communicate with other SDI-SCAM agents—each agent has the ability to communicate and exchange information with other agents (although the content of this information and the agents with which it is shared may be controlled, as will be discussed later). In normal use, a remote agent will send filtered traffic information downstream. When other agents detect potential security threats, warnings will pass upstream back to the remote agent.

3) The maintenance of various protections—On a continual basis, SDI-SCAM agents send and receive warnings and potential countermeasures relevant to whatever network risks are the most likely at a given time. For example, if a computer virus is detected at one node on the network, the local agent will immediately communicate a warning to all other agents in its contact neighborhood. If an attack is especially bad, the agent will have the ability to swap into the backup memory or contact other agents through alternative communications lines.

SDI-SCAM can operate either as a standalone security system, or as an additional layer that subsumes (and takes priority over, in cases of conflict) existing security protocols.

4) The ability to repair damage—Even after a node is known to have been attacked, the SDI-SCAM agent can be given access privileges such that it can aid the system administrator in controlling and repairing whatever damage has resulted.

5) The ability to scan collected data traffic for patterns consistent with threats—In many configurations, SDI-SCAM agents share their traffic information with a dedicated SDI-SCAM server capable of gathering and sifting through the entirety of the traffic data in order to detect

patterns consistent with a network attack, be it related to a hacker or to a new virus. Certain traffic events, which individually may be mistaken as simple anomalies, may become more apparent when the totality of a network's (or multiple networks) traffic is considered on a macro scale.

6) Notifying system administrators in the event of certain probabilistic attributes exceeding certain levels—The system's implementation of a Belief network (as herein disclosed) may also be used to determine under what overall conditions of probabilistically determined and descriptive variables it is advantageous to notify the system administrator. These variables can be based upon the predicted likelihood for the system to solve the problem, prevent certain types of problems, undesirable events and/or quantified degrees thereof from occurring or manual/or manually adaptive rules may prescribe threshold settings for some or all of these key variables. Among other situations, the system administrator may be notified or alerted in cases in which patterns detected may be only slightly suspicious according to the standard screening methodology, however, are consistent with SDI-Scam's best estimated simulation model from its distributed agent sources of how a threat might emerge, e.g., by mutation and re-emergence, e.g., after initially being defeated by SDI-Scam.

Meta-data associated with the accessor like a watermark that can also be embedded in code that contains digital credentials of the user, however, incorporates the use of "potentially" rogue, irresponsible, or destructive individuals as per the types of associated predictive attributes from criteria as disclosed in a presently preferred embodiment. The code cannot be tampered with without interrupting the watermark. A more general term for this "invisible" code sequence, which appears random to a would-be interceptor, is "embedded code". Typically, the embedding is done in a much larger nonsense message to apparently random patterns (in as much as the application code would already be encrypted) and this nonsense message content may not be required. Also, it can be associated with functionally defined portions of the code, which pre-approve certain behaviors. The system could also be based upon willingness of the accessor and/or code which s/he writes to statistically pseudonymize and profile the user with that of the patterns/types, etc. of code s/he has written in the past, thus predicting even without explicit identification who is the likely author and what s/he is like, i.e., what is the statistical probability distribution of the individual to each of a variety of previously known identities based upon code morphological characteristics, functional behavioral features, human behavioral features (e.g., if it is accompanied by a human attack). Pseudonyms and resolution credentials may be useful to authenticate the basic intent and MO of the author of the code while use of cryptographically interoperable pseudonyms, i.e., multiple unique but single identity aliases which are linkable to that single author only by SDI-SCAM for its security analytical purposes and under prescribed conditions (as data disclosure policies) as dictated by that author. Pseudonyms may be used to insure the same level of anonymity of the author as uncredentialed code. This approach could, of course, either be implemented as a local protocol (i.e., existing applications, application updates and new applications could all possess these credentials verifying/certifying that the present code was written by an individual who has been certified by a trusted certification authority as being non-malicious). This approach and the above pseudonym based identity protection scheme, while applied in this case to the application of software security are disclosed in detail for the application of identity protection, data privacy and security from rogue individuals interacting on communication networks such as the Internet. These relevantly related techniques are well described in the parent case as well as in U.S. Pat. No. 5,754,938, entitled "Pseudonymous Server for System for Customized Electronic Identification of Desirable Objects".

Within a typical context, this type of code certification should be impervious to a "man in the middle" attack. Such embedded messages (or in a similar cryptographic variation, "fingerprinting") are inherently effective for the security application proposed inasmuch as any rogue code which a system attacker would attempt to insert into a certified application or communication or other communication containing executable code would contain within its sequences continuous portions which do not contain the embedded credential-based sequences. Likewise, in case the would-be man in the middle attempted to remove certain data, (e.g., credentials or functional application code) the fingerprinting technique would recognize the specific extracted code segments. This exact same problem can be solved alternatively another way in which the primary objective is to transmit data containing a message the existence of which is not possible to be detected by a would-be "man in the middle" attacker. In the example approach in which a content bearing message is embedded or fingerprinted into the application code (or less desirably in an associated larger message), the message can only be identified by the recipient (the local SDI-SCAM agent) who may also be similarly hidden or "steganographed" as with the originally sent message (in order to verify receipt of the message by the authenticated recipient. There may exist in this content bearing message a variety of useful credentials incorporated therein including but not limited to credentials approving both authenticity, untampered state and authentication of the sender and/or author as well as proof of certified "good intent" on the part of the code author. The technique for insuring that the embedded sequences are completely undetectable, while at the same time being diffusely spread throughout the code is typically performed by using encryption techniques (e.g., pseudo-random sequences) to determine the positions of the sequence bits within the remaining code in order to thus pass a message to the recipient (the local SDI-SCAM agent) containing the credentials and potentially the message of the coordinates of the associated meaningful sequences, such that all of these content bearing sequences appear among the remaining code as random noise, including the portion of the message containing the encrypted coordinate data of which coordinate bits possessing the totality of the embedded or fingerprinted message can be found within the application. Alternatively, this message containing the coordinate locations of where to find the meaningful bits containing the content bearing message may be embedded within a larger message which itself appears to consist entirely of noise (which in and of itself lends the security of the embedded or fingerprinted message contained therein). The primary hurdle in this case is to enable the recipient to be privy to certain data, which is not known to a would-be "man in the middle" attacker namely where to look for the message, i.e., the coordinates of the meaningful data constructing the message. This "shared secret" between the sender and the receiver could be conveyed to each party initially by a (one time) physical distribution (e.g., contained within an application if it is physically distributed, such as on a disk, or visa vie the OS or CPU, etc. In one variation in which the dissemination of this message needs to be performed on a network wide level (or group level), the shared secrets may be physically distributed, once to all parties in a group and, subsequently, all parties would be able to instantly initiate communications with the security guarantees achievable through the presently proposed methodology.

Finally, it will be sufficiently obvious to one skilled in the art that the presently proposed methodology has numerous potential applications in cryptography and data security and thus the means for distributing data coordinates to a recipient of a steganographed message for conveying (and if desired reciprocally confirming) a message is in no way limited to messages, containing credentials and authentication certificates about an author and/or sender. For example, the present technique could be very prudently employed as a means to distribute and replenish shared set keys within the context of U.S. Pat. No. 7,350,069. It may also protect against man in the middle attacks against distribution of private keys in Pki protocols.

SDI-SCAM Network

There are multiple network morphologies possible. Major configurations include the following:

1) Local network: SDI-SCAM enabled machines may form a local network, such as a LAN or WAN. Gateways to external networks (such as the Internet) can be fully controlled through SDI-SCAM enabled routers.

2) Open network: On the other hand, SDI-SCAM enabled machines can be connected directly to outside systems (such as a desktop system connecting through a generic ISP), but which maintain communications with a chosen neighborhood of other SDI-SCAM enabled machines.

3) Centrally organized networks—In this configuration, thinner SDI-SCAM agents are placed on individual nodes; these agents continue to be responsible for direct security and repair, but transmit gathered traffic information to central SDI-SCAM servers containing dedicated hardware and software capable of swift and very in-depth analysis of the gathered information.

4) Distributed networks: In this configuration, each SDI-SCAM agent shares the responsibility for traffic data analysis and the generation of preventative measures with other agents. A peer-to-peer morphology would work well in this case.

Inter-Agent Communications

Although there is clearly a benefit for agents to fully pool all information, it may be desirable to control both the content shared and the partners with which a particular agent is allowed to interact. These parameters can be set at the local level according to users' preferences.

SDI-SCAM agents may in fact negotiate with each other depending on the value and sensitivity of particular information, as well as the value of any likely synergies between them. Multiple agents may meet in virtual information sharing marketplaces.

Another level of security can be gained through the exchange of obfuscated, but still valuable, information. Such randomized aggregates would allow systems to share fundamentals without revealing details of their particular data (for example, agents could share times of attempted log-ins without revealing the associated user ids and failed passwords).

In more complex realizations of this system, associated groups of agents may form coalitions, with information shared freely internally, but shared with conditions externally.

A further feature is that communications between agents need not be perfectly symmetric—in other words, different agents may send and receive different sorts of information. This might apply, for example, to a centrally organized SDI-SCAM network: outlying agents would have no need to transmit detailed traffic data to each other, but would rather transmit it directly to a central server. The central server might communicate with other central servers, in which case it would transmit high-level information relevant to the processing of the entirety of the traffic data; on the other hand, when communicating with outlying nodes, the central server might only transmit simple virus protection instructions and metrics which are substantially devoid of any data which suggests what types of information, attacker strategies or applications are running on other nodes on the system which are outside of the network of nodes and which are currently trusted by the nodes from which the centrally collected and processed data had been acquired.

Furthermore, there may be an additional or alternative approach to guaranteeing absolute data security at a local network or machine level while enabling maximal or complete harnessing of all of the statistical knowledge, which is present across the entirety of the network. In this approach it may be possible to operate SDI-SCAM or certain particularly sensitive portions of it with its multiple agent architecture as a singular trusted, yet distributed multi-agent system. In this variation, all of the locally performed or assigned agent functions are assumed to contain sensitive data belonging to external third parties and thus all processing activities, data communications with other agents or the central SDI-SCAM server occurs within a secure trusted and untamperable environment such that the only knowledge ultimately accessible by any given agents, associated local server or network on which it physically resides may be the collection of executed functions which are performed by the local agent on behalf of the SDI-SCAM to protect the local system as herein disclosed.

The order and way in which agents communicate with each other may be highly conditioned on the particular nature of a given system. Criteria include (but are not limited to) the following:

- overall vulnerability of a system;
- importance of the system to the integrity or functioning of a network;
- sensitivity and value of the data stored on a system;
- probability that the system has already been compromised or damaged;
- characteristics of the network traffic going to and coming from the system;
- overall importance of a system to a potential or identified hacker or specific system subcomponent.

This may dynamically change from moment to moment and is predicated by a probabilistic estimate determination variable of the intruder, whether autonomous or human and/or by human expert based estimates who are ideally familiar with local competition (or enemies) and broad knowledge of what types of knowledge on the system would be most of interest to which other entities or individuals and for what reason. If an individual is specifically identified this statistical model may further borrow and integrate techniques disclosed in co-pending U.S. patent application Ser. No. 11/091,263, filed Mar. 26, 2007.

Updates and communications between agents (termed "polling") may be based on schedules or on circumstances. For example, a remote agent may be updated with new antiviral software once a month; however, if any other node on the network is attacked, the schedule is suspended and an immediate update is performed. Certainly even if an attack which, for example, has only begun to occur or which has not even positively been confirmed as yet, triggers SDI-SCAM's system alert feature, other nodes on the network most preferentially/urgently those which are physically proximal or in other ways similar may also be put on alert status and SDI-SCAM's repertoire of protective features may be triggered so as to begin operating at a heightened level of defensive activity. As indicated, there may be a range of different system defense levels corresponding to a decreased probabilistic likelihood of a threat and the likely severity thereof should this threat exist. Local system administrators are notified appropriately as well. Determining the likelihood that a threat upon a particular node or network will also be carried out against any other given node can be predicted by such variables as commonalities at an organizational or strategic level, data communication occurring there between, commonalities in the existing or perceived data on applications contained or functional objectives achieved upon that node, presume interest level that a potential intruder of the attacked node or network may also have with the other node, etc.

Polling priority may be based on calculated likelihoods: for example, if various factors indicate that the probability is high that a remote node has been infected by a particular type of virus, the central server may be put into immediate communication. Polling priority will also depend on the nature of the nodes and the way in which their agents have been seen to communicate. U.S. Pat. No. 5,754,939, entitled "System for Generation of User Profiles for a System for Customized Electronic Identification of Desirables Objects" may be used as the basis for optimizing the way in which polling is performed.

Illustration

FIG. **1** provides an illustration of some of the configurations discussed here.

Analytics

Given the number of different security objectives, as well as the number and diversity of possible agents and network configurations, a fairly broad range of analytical tools are employed by SDI-SCAM. They include, but are not limited to, the following major categories of analysis:

Methods to Detect and Classify Direct Intrusions

Direct intrusions are attempts by unauthorized entities to enter a particular system, either over a network or through local terminals. These can range from fairly unsophisticated attacks (for example, teenage "script kiddies" using standard public domain software to scan for open ports across a list of target IP addresses), to extremely skillful attacks that are focused on a very particular target (as might happen during corporate espionage).

Since SDI-SCAM agents are able to dynamically monitor and analyze as well as control all in-going and out-going traffic, they are in a good position to detect and counteract such attacks.

1) Attack Patterns Consistent with Previously-Observed Patterns Across the SDI-SCAM Distributed System.

Each SDI-SCAM agent has access to a shared database that contains the signature patterns of previously observed (as well as verified) attacks. The likelihood of these events having been actual attacks may be probabilistically estimated so as to optimize the precision of SDI-SCAM detection/diagnosis as well as countermeasure deployment system modules. Such patterns might include the use of a particular password list, log-ins at particular time intervals or frequencies or times, log-ins from suspect IPs, (and/or combinations thereof) constitute a few of the straightforward examples.

If such a pattern is detected, the resident SDI-SCAM agent may opt to deny all entry to the IP of the incoming log attempts, or it may opt for a more sophisticated defense, such as opening a "honey pot" trap, a virtual space that simulates the environment of the system that is being protected. The hacker, believing that he has actually broken into the system, can then be monitored by SDI-SCAM, as his behavior might give clues to his location, identity, and motives and incriminatory evidence, if desired. Assuming the hacker has learned (or possesses) enough knowledge about the system to detect "honey pot" traps it is advantageous and precocious to possess at least equivalent knowledge regarding SDI-SCAM to possess at least equivalent knowledge regarding its own environment and to be able to enable the system administrator access to that knowledge as well as (via SDI-SCAM) knowledge known or suspected to exist within a probabilistic context regarding the hacker or threat and its strategy and/or this knowledge may be acted upon appropriately by SDI-SCAM in automatic mode. Invariably all counter measures (such as honey pot traps) used by SDI-SCAM can be used to the advantage of the hacker if s/he is aware of the strategy of SDI-SCAM to monitor, model, locate in order to ultimately catch him/her.

2) Utilizing Data Modeling to Adaptively Learn and Recommend Appropriate Countermeasures

Implementation of practically viable automated countermeasure scrutinization and recommendation scheme is quite achievable:

a. If the conditions/parameter triggers are simple and unambiguous, and b. If the system administrator is notified and able to intervene while exploiting the system's analytical knowledge and system-generated recommendations and scrutinies by the system on behalf of his/her chosen response decision.

In the ideal scenario, because rogue attacks are capable of performing increasingly effectively against system security protections (in addition to being more sophisticated and expeditious) and especially with regards to leveraging the system's own abundantly capable resources, it may be ideal as a complementary measure to building redundancy into the system resources in the interest of expediency of decrypting a counter measure, to also immediately respond in automatic mode, then solicit the active, albeit system-guided intervention of the system administrator whereby more significant decisions can be perhaps more confidently and prudently executed (e.g., whether or not to delete potentially corrupted files/portions of system data at the server or network level), whether to guarantee a certain portion of the network but allow certain essential functions to continue for the time being without code exchange, whether or not to attempt to infect the hacker's machine (or analysis code into the virus itself) which may provide additional detailed information as well, etc.

3) Novel Attacks

In some cases, attacks will follow completely new or novel patterns.

Such attacks can be detected in different ways. One solution is to configure a Bayesian network to constantly gauge the probability of an ongoing attack by monitoring network traffic activity (this configuration can be done by human experts and/or through machine learning techniques). A variety of factors can be extracted from the network traffic across all SDI-SCAM agents in the local network—for example, the number of failed log-ins, the identities and IP addresses of those users attempting to log in, the importance, sensitivity or "value" (more specifically "perceived value") of particular target files or contents potential adversarial entity or prospective hacker, etc. These factors are fed into ongoing probability calculations, which may trigger a system-wide warning if a certain threshold is surpassed.

Keystroke monitoring virus must be mentioned since it is impervious to NORTON™, etc.

For example, suppose a ring of corporate spies tries to hit a company's network simultaneously. SDI-SCAM agents across the network will report the use of unauthorized passwords originating from the same IP or IPs to which associations have been constructed via SDI-SCAM based upon historical statistics if the probabilistic likelihood of such events occurring independently might be so unlikely that the Bayesian network would immediately increase its estimate of an ongoing attack.

4) Attack Warnings

Note that in all cases, when an attack is suspected the resident SDI-SCAM agent will immediately alert all the other SDI-SCAM agents in its network neighborhood, sharing all traffic information relevant to the on-going security issue. Such warnings will include information related to the particular nature of the problem, in particular the probability and nature of the threat (for example, communication with an unsecure system, access by an authorized user, reception of potentially infected files, etc.).

When an on-going attack is announced, SDI-SCAM agents receiving this information may opt to increase the security levels of their own systems. For example, users may be required to telephone at the time of their log-in to verify location (through caller ID) and voiceprint.

Methods to Detect and Classify Viruses or "Trojan Horses"

Origins, possible paths of transmission across sites, etc. types of files (e.g., particularly vulnerable or vulnerable origin site), may be analyzed to provide ideas as to how to use this data to make a vulnerable application, Trojan horse attempt impervious, make rogueness, crypto query, even rewrite code.

Another vector of attack is through viruses (which are often unauthorized and malicious programs attached to files, email, or documents) and Trojan horses (seemingly innocuous programs that contain hidden programming capable of causing damage).

Code Analysis

The conventional viral detection methodology is to scan the code (in the case of executable modules) and macros (in the case of smart documents, such as those generated by Microsoft WORD™) for patterns that have previously been associated with viruses or malicious programming.

SDI-SCAM maintains up-to-date records of all known viruses and checks all incoming files (and periodically, all stored files) against these records. A match indicates that a file is potentially infected—the user is alerted of the danger and automatic defensive measures may be set into motion.

Behavioral Analysis

SDI-SCAM monitors all processes for behavior consistent with viral infection. For example, a program that is observed to open and modify a wide range of heterogeneous files, which accesses the mail system's address folder, which aggressively propagates copies of itself, which engages in recursively redundant actions whose objective is designed to achieve no useful purposes or frequently which aggressively/repetitively generates or obtains data files in order to propagate inordinately voluminous and/or large files (possibly including itself) resulting in bursts of traffic (thus overloading valuable network transmission capacity), which performs similar recursively redundant actions resulting in consumption and overloading of valuable processing capacity, which modifies or mutates its own code (and/or behavior), or which opens unexpected communication ports with outside entities will be flagged as a potential threat. Unquestionably, SDI-SCAM's highly distributed data traffic monitoring and behavior and code analysis facilities as a combined approach give it a marked and compelling advantage in rapidly analyzing those behavioral patterns and characteristics most commonly associated with a rogue code such as viruses, Trojan horses, worms, etc. whose tell-tale signs could not be identified nearly as expeditiously as that of SDI-SCAM's distributed agent monitoring architecture. Such commonly occurring signatures which SDI-SCAM's distributed Bayesian methodology is particularly well suited includes those patterns of self-replication and dissemination through address books, email, web browsing sessions, etc., as well as the co-occurrence of identical or related patterns of behavior and code sequences in conjunction with these replicating and self-propagating patterns as observed only on a network level. Certainly part of this behavioral analysis may encompass attempts by SDI-SCAM to classify the identity or type of virus based upon all of the above observed characteristics as well as attempting to extrapolate its high level objectives and associated executable rule sets based upon its behavioral patterns associated with the conditions/variables of the environment which it has encountered, the data which it has likely accessed, the actions, events and countermeasures to which it has been exposed, the code within which it has likely been embedded, etc.

Although it may be difficult to delineate rogue from innocuous code it is certainly within the scope of capabilities of SDI-SCAM to utilize all of the available data, both behavioral and code sequences, in order to attempt to reverse engineer the code for the purposes of both predicting its future behavior, likely past behavior and high level objectives. For example, SDI-SCAM could replicate the code inside of an environment which is quarantined from the network, but which is a replica of the network or a portion thereof. SCI-SCAM could then monitor how the code behaves in this simulated environment to the actual one as well as observing its response to targeted stimuli, which may, for example, provide opportune conditions for the most likely rogue actions to be performed. This analytical procedure may be performed in response to a predicting statistical model (designed to predict the code's behavior) when a decision tree could be used to dynamically select the set of functions to be executed which based upon the same model are correlated and then predicted to elucidate responses on which are the most optimally revealing, reveal the most revealing which is needed to complete the construction of this data model for the codes for being able to predict the code's behavior across a wide array of conditions, actions, software and data to which it may ultimately become exposed within the entirety of network(s). In depth analysis of potentially suspicious code although challenging as it may be could potentially provide system level insights into how to best respond to the potential threat and if mandatory the nature and aggressiveness of countermeasures to be taken or recommended to the appropriate human system security counterpart.

The user will be alerted, and if he confirms that the program is operating outside of expected parameters, or if the user does not recognize the program, it is taken offline until it can be examined in detail by an expert.

Dead-Ringer Analysis

Although not currently a threat, it is likely that infectious programs will be able to simulate the behavior of human users. A suite of behavioral response tests can be developed to detect and counteract such entities, e.g., a probabilistic model based upon other previous threats in the statistically similar characteristics (including behavioral characteristics and certainly those determined to be the most likely to be the same). Queries which may be required of the "user" to be answered correctly or to perform a task (e.g., compose a block of text on the basis of a query) in order to proceed could be solicited of the user which are crafted such that an emulating virus would likely fail such query procedure. Moreover, Natural Language Processing methods can be used to analyze outgoing text for irregularities consistent with a non-human origin. It is possible that in a similar fashion, that, in theory very smart emulations of existing code could be manually or even automatically on the fly created which emulates in many respects existing "good code", but which actually is designed for malicious objectives or, for example to take over control of the good code or replace it with the rogue version. As additional attributes of the system, the system may determine probability and degree of ill motive of individuals of most likely suspicion (if such suspicion is high enough to be of reasonable concern). Typically, common suspicion of particular individuals can be linked to unscrupulous employees (present or former), disgruntled employees, disgruntled spouses of key persons/owners (e.g., changing files, information release, etc.) to embarrass or defame the person or to feign a verbal or tactical attack on a friend, associate or colleague. Such "suspects" could also include trusted partners who may be confided with knowledge of the existence of unique information which could be of interest directly or could even help or strengthen that party in its business position with its "trusted" business partner.

Control of Triggers

If the probability of an infection is deemed to be high, SDI-SCAM may control the generation of events that could potentially trigger the reaction of a resident virus. For example, if a bank suspects that a corporate virus has infected the system, all transactions may be suspended until the virus is cleared. Otherwise, the action of a user requesting an on-line transaction (thereby releasing his personal password to the system) may trigger the virus into capturing and re-transmitting his personal information.

Tracing Threats Back to their Original Source

In traditional system security techniques this objective is highly desirable and yet extremely difficult. Nonetheless, SDI-SCAM's functional features lend themselves quite well to the design of certain particular types of applications, which can be useful in addressing this particular problem. For example, the following example applications may be herein considered:

1. "Infecting" the Hacker's Machine (or the Virus) with a Virus, which Logs and/or Conveys Back to the SDI-SCAM Agent the Location, Behavior, Files Infected as Well as all IP Addresses of the Machines in which these Files Reside.

This approach is likely to work provided that the implanted virus by SDI-SCAM is not recognized by standard virus scanning software or other IDS systems and assuming that the receiving machine is not programmed to block any outgoing messages. Thus, the success would be determined in part by the effectiveness of the virus to take control of the adversary's (or rogue virus containing) machine. This type of direct analysis will both enable preemptive alerts of exactly where the virus may be spreading to other machines and/or networks as well as provide valuable statistically confident data as to the function, behavior, data or code affinities and behavior in response to infection of the same as well as epidemiological characteristics which could be extremely valuable as to anticipatory determination and qualification of the associated threat on other machines, as well as the most appropriate countermeasure each local agent should implement or receive in response. Certainly, this approach could be useful for viruses, which possess particular rapidly proliferating characteristics, rapid infliction of destructive behavior. For example, one could imagine the behavior of more sophisticated viruses which might proliferate themselves as redundant messages so as to rapidly overwhelm network capacity and/or memory processing and/or implement parallel strategies.

This approach could also enable SDI-SCAM to model not only future epidemiological characteristics of rogue software but also that of post epidemiological behavior (which machines or networks were likely to have been infective previously based upon presently known epidemiological characteristics) and the devices/networks which are known to be and probabilistically suspected of being infected by the same virus (or mutated variant thereof). Certainly reconstruction past, present and future behavior in this regard could be relatively easy to perform for worms that may require access to ISP server logs for other variations which may use email and web server connections as a medium of transmission. A protocol also may allow for the existence of a latent tracking virus to reside within all machines which can be, in the case of significant probability of a threat in and among a network community or otherwise "group" an excessive probability of a threat, the tracking virus may be remotely activated by a multi-casted activation message originating form a core (or root) server.

2. Use of SDI-SCAM Architecture for Application Level Security

It will be increasingly important in the future for many of the functions of SDI-SCAM as implemented within the context of its presently disclosed distributed statistical analytics to be implemented not only at the level of a distributed network security system but also at the individual application level. That is to say that SDI-SCAM agents could, in addition to the above described system level implementations, also implement their various functions for data collection, analysis, and countermeasures at the application level as well both to implement other application level security protocols as well as incorporate into the statistical analytical scheme probabilistic attributes regarding the behavior functions, etc., of such rogue code within the context of the particular relevant applications in need of protection, albeit using the same distributed adaptive modeling and countermeasure response protocols described herein in comprehensive fashion.

Methods to Detect Tampered Files (Semantics and Content)

It is sometimes the case that intruders, rather than destroying or removing files, will simply alter them in potentially malicious ways. For example, students may attempt to hack into their school system in order to change grades, or a more advanced hacker may attempt to break into a bank to fraudulently increase the balance in his account, into tax or criminal record databases in order to change tax liabilities, records of property ownership or criminal records, into professional board's databases in order to change licensure status. Similar tampering may occur to files whose contents may relate to the hacker (e.g., employee files of present or past employers). Malicious code may, in theory, perform all of the functions that a human may perform, perhaps, however, potentially even more unobtrusively and elusively in that it may be more difficult to trace and flag than a human if the code is very small, robust and capable of focused but sophisticated emulations of legitimate applications and users.

In addition to the above suggested techniques for use in tampering detection and ultimately prevention (or even

tracing the origins of tampering attempts), there are other straightforward IDS-based approaches by which such attempts could be countered (and could even complement the above safeguarding scheme, for example, in terms of being a default detection scheme and/or in corroboration of the presumed integrity of credentialed individuals). Thus, the following IDS-based alternative technical approach is also provided as well. The local SDI-SCAM agent maintains logs that detail the general characteristics (size, word count, hash code) of all documents on the system. The time and circumstances of any changes are cross-checked against average traffic patterns for that particular system. Hence, school records altered at 3 am (in a district where all school secretaries worked strictly from 9 am to 5 pm) may be flagged as potential objects of tampering.

Tampered files will sometimes show a marked change in writing style or technique. Natural Language Programming (NLP) techniques may be used to detect such changes. Certainly in the event of these suspicious activities and other conditions, it may be advantageous to retain not only the associated statistical data (as the SDI-SCAM does automatically) but also details regarding the events. This could, for example, be later analyzed by humans to compare with other similar suspicious patterns also captured in detail in order to attempt to identify patterns, more subjective signatures, or hall marks which may not be able to be performed automatically (such data may also be useful for potential legal evidence).

Methods to Detect and Classify Untruthful Commercial Messages

Untruthful messages represent a more traditional kind of deception—the technology of the delivery is not damaging, rather, the content of the message itself is untruthful and may prove harmful if taken at face value by the receiver. A good example of this is the "Nigerian Scam," a widely disseminated email that purports to be authentic, asking the receiver to give the sender access to an American bank account in exchange for great financial reward. The result, of course, is that the receiver ends up being defrauded of large amounts of money.

1) Cross-Checking Content Against Known Hoax Documents

SDI-SCAM maintains a database of questionable messages and uses natural language programming-based techniques to compare incoming messages with previously logged deceptions. Thus, when a suspicious message is detected, the receiver may be sent a secure attachment by SDI-SCAM with an email stating that there is a high probability that the mail is untruthful, and indicating pointers to web pages that discuss that particular deception. If a user is nonetheless deceived by such a message, the local SDI-SCAM agent may be alerted. It will transmit the text of the novel message to a security database, allowing every other SDI-SCAM in that network neighborhood to be aware of the danger. In such a case, the agents may retroactively warn users by scanning old emails and alerting receivers of possible deception.

Certainly in such an event, autonomously implemented counter measures may also be performed if appropriate as a defensive or evasive action or deterrent, e.g., if a pass code was inadvertently sent out (and it was not blocked by the system) the pass code could be automatically changed or temporarily frozen or if a personal bank account or credit card number were sent out in a suspected inappropriate context (again assuming it was not blocked at the source by the system), the account could be automatically temporarily frozen and the number changed or (for example) the account automatically set up as a honey pot trap to acquire just enough information about who the suspect entity is in order to catch him in an inappropriate act of fraud or deception.

2) Predicting Possible Hoax in Novel Message

In cases where a message is not closely correlated with known hoaxes, it is still possible to analyze (using natural language processing techniques that are currently well known to the art) the content of the message and flag any suspicious content:

the content of the message can be cross-checked against recent news stories discussing hoaxes; and.

the names and return email addresses of the incoming mail may be checked against those of known hoaxsters.

Automated semantic analysis of the message may be performed for language consistent with persuasion or appeal to greed (or other weaknesses). This analysis is performed on the basis of adaptive rules which may be updated with feedback.

The identity and personal profile of the receiver may be correlated with the characteristics of known victim groups. For example, messages sent to rich elderly individuals may be given additional scrutiny.

The purported identity of the sender can be checked against the path of the email. For example, a message claiming to be from the IRS should trace back to an official government system.

A probabilistic assessment of the likelihood that the sender is fraudulent may be performed through a modified version of the system described in co-pending U.S. patent application Ser. No. 11/091,263, filed Mar. 26, 2007, in which the system's probabilistic determination of predictive attributes relevant to an association with fraudulent, unscrupulous or disruptive behavior (in an on-line context) is performed—of course, the sender if self-identified may also be fraudulent. The on-line sender just prior to the first receiving node on the system may also be analyzed which is a reasonably reliable tracking means if SDI-SCAM is a ubiquitous protocol (e.g., for patterns of being the origination node for previous problematic messages and/or the techniques disclosed in the same co-pending patent application), whereby the system may probabilistically predict the suspicion level of an individual(s) or organization(s) associated with that sender as being linked to other scams and/or other illegitimate or questionable activities. Related techniques may use other advanced customized semantic analysis and/or adaptive rule based/statistical techniques (preferably in combination) in order to estimate the degree of potential harmfulness of the content.

Because SDI-SCAM's distributed agent and sensor network perform consistent detailed surveillance capable of monitoring across a diversity of different heuristics, it is possible to monitor for early patterns which are indicative of an attack which may be quite sophisticated either of a large-scale nature or which targets a specific type of system and therefore which may tend to go undetected (e.g. visa vie its hardware, operating system or possibly software) but may be destructive to very important systems like control systems for critical infrastructure. These detectable patterns may be based upon experience of the agents (to previous known threats) and unknown threats e.g. detecting patterns which are not normal but not definitively linked to threats of a previously known nature. Certainly fuzzy probabilities analyzing of combinations of these condition states will be able to fine tune triggers for potential alert states which are matched appropriately e.g. balancing potential threat to the possibility of variations of normality to the real degree of threat and given the degree of importance of early detection

and response given the overall importance of the systems at hand and the predicted virulence of the threat that the detected pattern presages.

Thus with regards to specialized systems and networks, certain specialized patterns corresponding to the type of end result that the attacker may like to achieve may be preprogrammed into the system even if such novel attack had not occurred previously. If there were infections of specific types of operating systems or software or hardware which also corresponded to certain of these components that existed in critical infrastructure it would be cause for concern that such a threat (e.g. a worm) may in fact already exist for purposes of attempting to target such critical infrastructure. This is one reason that redundant components of this type which may be placed on the network or even assembled into analogous system architectures may be useful in preemptively becoming targeted (as "bogus targets") for purposes of early detection and characterization of the threat before it actually targets its intended critical system. This is just one example of many techniques and strategies which can be used by SDI-SCAM to complement its ability to utilize its distributed sensors to identify and characterize potential threats either at the very onset or even before they occur or certainly as they are emerging and beginning to spread. Again the nature of the automated notification, additional defenses and/or remedial counter responses may be both targeted to the threat, to the targeted and/or vulnerable systems as well as to those where SDI-SCAM is able to predict the spread is most likely to occur, e.g., based upon commonalities of various sorts such as communication links, business/commercial affiliations or social profiles, etc. On traditional networks in monitoring for potential signs of large-scale attacks experiential knowledge must be of course heavily weighted. There are a growing variety of and new variations of such threats. However they tend to fall into general categories for which significant experiential data can be accrued.

The content may be corroborated with the content of known and trusted documents, e.g., through the use of content matching techniques. More elaborate extensions of this approach may include more advanced semantic analyses of the subject content with its credible and updated/current matching counterparts whose algorithms are custom configured to confirm (or alternatively flag) or assess the probabilistically estimated "truthfulness" of contents (where "truthfulness" may be reassured according to "confirmed with credible source" as well as scalar measures of degree of likelihood of untruthfulness if the source is unconfirmed or, for example, exhibits semantically suspicious inconsistencies with itself, with credible sources or other patterns which are consistent with fraudulent or deceptive material).

The system may also detect suspicious content, for example, if its appearance co-occurs in the same message with rogue code (for example) is co-located (in the same portion of content) as a macrovirus.

Methods to Repair Post-Attack Damage

In some cases, despite the security, a system in an SDI-SCAM network may be damaged by an attack. If the attack is completely novel, a human expert may be called in to fully analyze the situation and develop appropriate repair protocols. These can then be sent to a central SDI-SCAM damage-control database for use in future situations. In this way capturing as much data and statistical information regarding the attack and its historical counterpart is valuable both as analysis data for the human or to enable the system to construct its own optimal repair protocol.

If an attack method is not novel, the local SDI-SCAM system may access this same damage repair database for solutions to the local problem. Among the remedies to damage from an attack: users are alerted, suspicious files are deleted, backup files are loaded, and current memory is completely cleared and reloaded with an image from a pre-attack state.

Additional Embodiments

The inventors further propose a scheme in which a Server provides security and validity information about free online software that can be downloaded off the web. The Server keeps a score that is made up of three components. The first component is computed based on the security of the software itself. The second component is computed based on the experience users have with the program. The third component is based on the reputation of the programmer that created the program.

Component 1: The Security of the Program

In order to make sure that the program does not violate the computer onto which it will be downloaded, the Server will perform tests on the program the check if the program never accesses memory locations that it is not supposed to. Such tests will ensure that the program never reads data that is private on the client computer, and does not write to forbidden locations and thereby self-replicate itself. Our tests will also detect if the program has memory leaks using state of the art techniques, to ensure that the program does not launch a denial of service attack on the client computer in this manner.

Our testing environment will run the program in a sandboxed environment to detect illegal memory accesses. One of the novelties of our testing approach will be to generate test inputs with large coverage to ensure that many of the program's computation paths are exercised. We will do this using a two-pronged approach. Random testing will be used to catch problems in commonly exercised paths. However, pure random testing will fail to exercise the corner cases with high probability. To account for that, we will perform symbolic execution that constructs a logical property characterizing the exercised paths. The SAT solvers will be used to generate inputs to exercise paths that have not been explored. Finally, to detect potential memory leaks we will run garbage collection routines periodically. Since all our tests will be performed before deployment (and not on the client computer), the overhead incurred due to the sandboxed environment, the symbolic execution, and memory leak detection, will not affect the performance of the program when it is used by clients.

Digital Signatures

Once the software is checked for security it is vital to ensure that only this version of the program will be downloaded and not an altered version. Therefore, a digital signature will be created for the software. The digital signature can be stored on the Sever and used along with a public key to verify that the software is indeed the original one that was first tested. The signature is much shorter than the program itself so checking it is much more efficient than comparing two copies of the entire software. It is secure based on some cryptographic assumption in the sense that if some bits of the software are changed than the signature will not be valid.

Component 2: User Experience

It is possible that even with the secure checks that will be made in Component 1, the software could be hacked after the download. In order to detect such problems, in Component 2 the Server will keep a score based on the responses of users of how well the software worked for them and

whether they found any bugs or not. It is important to ensure that only responses of credible users will be used in the score calculation.

How do we Alert all Users if a Program has Complaints?

The approach we discuss here is based on a distributed agent-based model for network security as discussed above, which describes a distributed multi-agent system for real-time collection, monitoring, aggregation, and modeling of system and network operations, communications, internal and external access, coded execution functions, network and network resource conditions, as well as other assessable criteria. A Bayesian model is used to estimate likelihoods of various threat vectors. The model provides access to the reasoning behind its inferences. It may recommend or in some cases even implement responses to detected threats.

Component 3: Programmer Reputation

The Server will keep track of the identity of the programmer who created the software. Each time the software gets a good score from user's responses and from the security of the code the rating of the programmer will go up and this will be linked to all the different software that the programmer created.

There is an incentive for a programmer to create a user name that will be used for all the software that he creates. This way his reliability score is increased and his software will be more widely used. Even software that might be new that might not yet have a wide range of user responses in Component 2 will benefit from being linked to other software with high rating from the same programmer.

If a programmer chooses to hide his identity from the software then there will be no score for Component 3 for this particular software. Therefore, there will be incentive for a good programmer to associate his username with the program.

If a programmer chooses to identify himself with the software, then there are two possible ways to verify the identity of the programmer. One is to identify the software with an identity such as an email address or another created user id. With this scheme a user may create different user identities for each software application that he creates. In this way the software will not benefit from being linked to other software created by the same programmer. Therefore there is an incentive for an honest programmer to use the same user id for all programs. The programmer usernames will be handed out with a password so different programmers will not be able to obtain the same user-id. Therefore, a malicious programmer will not be able to link his software with the credibility score of another programmer.

A programmer that creates many programs will have further incentive to invest more time to create a Unique user id for which the Server requires paper identification such as a notarized copy of a driver's license. A programmer has the incentive to obtain such a Unique id because only one will be given to each programmer. Therefore, the programmer cannot create different programs that are not linked to this. If there is a virus in one of these programs then all the programs of this programmer will be given a bad score in either Component 1 or Component 2. Additionally, it is possible to trace the programs back to the programmer. Therefore, users looking for utmost credibility will look for software where the programmer identified himself with this Unique-id.

How do we Decide which Programs to Use?

A programmer may request to add his program onto the Server. Additionally, a user may request to add a program onto the site. The program does not need to be stored on the server. Only the signature can be stored on the server and as long as the program that can be downloaded from any site can pass the signature test it can be used.

Those skilled in the art will appreciate that this embodiment whereby a Server provides security and validity information about free online software that can be downloaded off the web can be implemented either as part of the SDI-SCAM system described above with respect to FIG. **1** or as a standalone scheme for improving computer security using the proposed credentialing scheme.

Whitelisting

This section highlights the value of leveraging the knowledge accumulated by the distributed agent of SDI-SCAM which particularly helps to overcome the limitations of whitelisting which in present day implementation is limited to a small fraction of the actual, but non-verifiable whitelist. Whitelisting particularly in the context of SDI-SCAM's probabilistic assessment of individuals, code, servers and user machines is a form of reputation system. Explicit whitelisting where a user is granted permissions to access certain servers or files or where they are considered safe individuals, machines, servers or code may, however, be used along with (or alternatively to) implicit whitelisting where such whitelisting functions are provided by SDI-SCAM based upon a probability distribution curve of safety or appropriateness of access (to typically specific programs, servers or machines) based upon matching or exceeding of a "safety threshold." The widely distributed agent based monitoring not only is capable of observing the large universe or network of networks but does it in a continuous and updated fashion. The non-whitelisted item would traditionally be "unknown" with the possibility of being unsafe or part of a black list in contrast to what is possible through the newly proposed paradigm in which what is not whitelisted has a very high probability of being unsafe.

Thus, in a global implementation of distributed agents, non-whitelisted machine and users, for example, could be for practical purposes effectively considered "unauthorized" in similar fashion to unauthorized "accessors" similarly for code or programmers in a closely monitored secure network. Compared to all known prior art counterpart systems, the use of a distributed sensor network to aggregate and provide means for data analysis and adaptive rules for modeling and predicting existing, emerging threats and threats which are yet to occur are particularly useful in early identification and prediction of the anticipated threats including anticipating severe threats (urgently warranting notifications and warnings), quantifying and/or characterizing servers, user machines, code, software and individuals regarding the present or predicted threat and revealing the likelihood of the threat (and/or potential severity or sophistication thereof). This includes the predicted safety or threat level, of a given user machine, server, program (including the user's own). Predictive threat level if high may also be revealed by black listing an item by implicit determination or if low whitelisting an item by matching or exceeding certain thresholds thus complementing explicit lists regarding the same with much more comprehensive data available through the distributed network of sensors.

In terms of one of its high level objectives SDI-SCAM strives for a comprehensive approach to higher efficiency computer security through the automation of these computer security processes. Thus, it is also useful to consider how it may be possible to improve not just the clarity in identifying and characterizing attacks and rogue activity but also how such information may be used to practically enable more targeted and forceful defense measures and counter offense responses in such a fashion that because the actions taken are not based upon the assumption of the possibility of unsafe or malicious activity but rather a high degree of confidence if not certainty of such activity or intent. While many types of defense and counter offense measures exist and are well known in the art, there is nonetheless a fundamental tradeoff between implementing such defenses or counter offenses for the sake of improving overall security and/or creation of a deterrent and the overhead of implementation in combination with (most importantly) the additional time/effort and overhead navigating through those defenses and in the case of counter offenses the risk associated with launching the counter offenses against a non-malicious individual or machine. Because higher value servers that store the data tend to be the target of the more determined and skillful attacker, stronger defenses and counter offenses must be implemented for more valuable information and machines. Still, for the most part, these measures cannot be targeted because of the indefiniteness of the individual or program until it is too late and the attack has already occurred at which point typically even a counter attack is no longer possible.

As indicated, there are numerous types of defenses and counter offensive measures and it is clear that more targeted measures as a result of more definitive and confident identification and characterization of a true threat before the fact of actual attack will enable greater efficiency in protecting against or deterring threats and at lower cost and overhead. In the case of counter offensive measures there will be also much less collateral damage. Moreover, for this reason it will potentially enable much more common use of deterrent measures (typically counter offenses) be they human or automated by significantly revealing and characterizing the threat and leaving little question as to the intent to cause harm or steal information before the fact of the attack. Again, while there are numerous examples of how more common use of these measures under the widely distributed agent paradigm (ideally for these purposes an internet standard) there is listed a few examples of defensive and counter offensive measures below.

2. A family of security software which scan for software vulnerabilities and recommend security patches such as Protector Plus for Windows and Securia Personal Software Inspector. Clearly observational data generated across a large number of installations of any given software using distributed agents deployed to aggregate data, leverage relevant variables in a probabilistic model, test vulnerabilities in practice and develop probabilistic assessment of determined likely vulnerabilities would do a more effective and comprehensive job in the software vulnerability assessment, particularly identifying the more subtle or hidden vulnerabilities. Data mining and recommended testing protocols visa vie human in the loop approach is likely more effective than a fully automated analytic approach in this application.

2. Vulnerability exploitation prevention software, e.g., Malwarebytes Anti Exploit tool and ExploitShield—clearly by determining a more comprehensive profile of what existing vulnerabilities are present along with their characterization it is possible to better protect against their exploitation through a more targeted defense scheme. Again, the analysis of widely aggregated data and testing for each potential vulnerability is similar to that of item 1 above.

3. Performing direct counter offensive measures against an attacker—As indicated there are inherently significant risks of inadvertently attacking a machine which is not a threat, for example, by forgetting to add that machine to the white list. Even if the risk is small users will see the potential

downside is greater than the upside. They typically will use instead basic encryption to further protect the file. In the distributed agent environment, observational data will allow the agent to recognize certain machines (e.g. via their machine code) as a non-threat (whitelisted by implicit versus explicit means) through their patterns of usage, e.g., files which tend to be shared (or similarly accessed by) whitelisted machines particularly those with higher security access permission will indicate at least the strong possibility that the accessing machine is not a threat and should not be counter attacked, but instead further investigated.

Another measure which potentially alleviates the risk of “collateral damage” to (mistakenly) non-threatening machines is to not destroy data or applications on the presumed adversary but rather to simply disable the machine, for example, using a counter offensive program which invokes the disabling function via the secure operating system. While a destructive virus, e.g., which wipes clean the hard drive is a fairly strong potential deterrent, so would be a program which locks the operating system until/unless the suspect attacker can prove to the owner of the targeted machine that the suspected intrusion was a “false alarm,” that no such presumably intentional unauthorized access had occurred. If the legitimacy of the “intruder” can be verified to the target machine owner, a remedy (or vaccine) may be provided to the previously presumed attacker’s machine. For example, it may involve the owner possessing a vaccine needed to eliminate the virus or to unlock the operating system which the virus had locked, which is, metaphorically, an “antidote” to the harmful side effects of an aggressive drug treatment. Thus, while the counter attack would be as harmful to the would-be attacker as a traditional destructive virus, it would be potentially immediately reversible and innocuous to a mistaken attacker (perhaps a legitimate intended authorized party).

In short, what is proposed is a counteroffensive measure (which may be but not necessarily a Trojan) which upon infecting or otherwise accessing the attacker’s (or unauthorized accessor’s) machine invokes the operating system to shut down. The victim of the original presumed attack holds a vaccine to reverse the operating system shutdown which typically requires that the original attacker prove that the access was legitimate or authorized and/or for the victim to verify that no harm was done on his/her machine or network. The privilege to use the code or tool to disable the operating system may be revoked by the OS provider if abuse is detected (e.g. using it as a mainstream attack strategy instead of a counteroffensive one).

How the direct counter offensive measure is implemented typically requires entry into the attacker’s machine. SDI SCAM if implemented on an ISP server local to the attacker or even in the form of a virus which infects the attacker’s machine may be able to learn about the attacker, his preferred targets, groups or other contacts or hackers which he may be directly or more loosely affiliated with will allow not only a detailed “snapshot” of the attacker but also the otherwise hidden activities, targets, timing patterns, etc., of him and his affiliates (if he does not work alone). This information may indicate whether the attacker is interested in data, and what type (s), and/or software and what type(s) (e. g., is it for the construction of a botnet?). If certain software is of interest perhaps he can be lured and baited to upload a Trojan masquerading as a program of potential interest. The same can be performed for certain types of user and server machines and regarding certain associated content (different ways files can be executable to masquerade as harmless data files). Certainly, detailed activity logs uploaded to the SDI SCAM network could be used to help warn known targets (or types thereof), individuals, system administrators, other (local) SDI-SCAM agents, etc. of potential attack, the type of attack typical of that attacker, vulnerabilities which tend to be exploited by that attacker and/or as pertains to the particular type of target user. For example, certain types of honey pot traps unique to the attacker could be set up. Of course, this information is useful in implementing defensive and counter offensive measures and may be delivered to the SDI SCAM agent on each user’s machine. For aiding potential SDI-SCAM or locally deployed counter offensives such detailed reporting may include software and potential vulnerabilities associated with the attacker himself. Even if the attacker causes significant harm to such entities as governments, financial institutions, credit card companies and corporate databases through theft of valuable data (as opposed to software code) it is possible to create executable code which mimics certain file types, or even modify certain content or file formats with latent executable functionality, e.g., Word or Excel. Such files can be completely dormant (non-executable) and thus innocuous if within a “friendly environment” e.g., an authorized machine code of the accessor’s machine is recognized or it may be detected as a potential threat or alternatively eliminated by the local virus scanner which may be tied into the SDI-SCAM system which in this case is fully aware of the fact of the file being used as “bait” for attackers. SDI-SCAM may have planted it as part of a targeted and timed counter offensive thus keeping accurate record of adverse threats and friendly threats as part of the overall and up to date status of existing and likely behavior of threats as well as the success statistics for certain defensive and counter offensive measures in view of certain types of sophisticated attacks and attackers.

Therefore Trojans (such as the above mentioned) as well as other approaches can be automatically recommended and implemented to targeted rogue machines in order to launch a counter attack to an intrusion and unauthorized access to data or software. Such files may be accessed from a honey pot trap or such Trojans may (again via SDI SCAM administration) be strategically selected and positioned within the network to maximize the likelihood of a likely attacker to upload the Trojan containing the SDI SCAM agent (technically-speaking, acting as a virus). For the counter-offensive variation, SDI-SCAM (or its human analyst counterpart) will tend to select or construct those files as Trojans within a given target machine which it deems most likely to be accessed (of interest) by the attacker given the total data profile collected about the attacker’s elicit data consumption patterns and relevance feedback from trial and error implementations which will indicate the preferred files based upon past attack or consumption patterns, the files and types of files snarfed by similar attackers or which perform similar types of attacks.

In summary, using SDI-SCAM’s distributed adaptive learning model not only defensive (as discussed earlier) but also counteroffensive measures may be recommended and targeted in response to a given threat or attack which is provided by the system based upon relevance feedback including trial and error from previously delivered responses to previous threats and attacks. Such relevance feedback may update the system’s adaptive rule base and utilize automated and human in the loop adaptive rules via data mining.

With regards to the state of the technical art of creating Trojans which appear to be innocent data files, currently, there is a significant gap between files and executable files.

To do harm a file has to 1) be executable and 2) be executed. The latter requires that someone click on it or otherwise run it. This is very unlikely from a hacker who has snarfed the file from a target machine since he has no real incentive to run a captured executable?

There are only a few types of not apparently executable files can be executed—mostly Word and Excel files which are running VBA. It would be tricky to get a VBA macro that was able to figure out it was on an adversary's machine which is black listed or non-white listed (where in a global SDI-SCAM implementation is effectively black-listed) and then attack that machine successfully. SDI-SCAM's distributed knowledge may help here. Also, possible modifications to the file format to automatically invoke an execution, determine foreign, unauthorized machine environment, e.g. mac code and other clues matched against a list of authorized machines or white list (e.g., stored in upstream database). If a typical destructive virus was used, the maximum likely penalty would be the black hat would have to rebuild his box, not a huge deterrent factor. Unless the penalty was the virus' disabling the OS until/unless the presumed attacker could prove to the data owner his in fact authorization and/or non-malicious intent. Once word got out about it the tactic would become less effective until/unless adoption became fairly prevalent. Of course, designing the Trojan to look like an innocuous (non-executable) file format is another approach. The problem of its likely hitting innocent bystanders, i.e. laptops not preauthorized for the file but which should have been (e.g. destroying a vice president's laptop because he forgot to have a file authorized for his box) can be mitigated (again) by the use of SDI-SCAM to determine the likelihood of machines to be authorized for access to certain files (even if not explicitly) authorized. The default approach is to only encrypt the files instead of initiating counteroffensive measures. This of course, does not provide a deterrent and encryption keys can be stolen e.g. through break-ins and compromised digital certificates. Nonetheless, one of the present SDI-SCAM objectives is adapting the defensive and counteroffensive strategy by better selecting the response and making the response more targeted, confident and appropriate to the target entity, its actions and intervening circumstances/conditions in view of associated probabilities of all of the above.

For example, depending upon the value of the data (to the owner and attacker), the present circumstances, etc., SDI-SCAM can utilize its probabilistic knowledge as well as data as to the foreign environment (e.g. non-white listed) to determine the likelihood of a compromise of the encryption key thus possibly warranting the executable code associated with the file to prevent decryption e.g. by self-destruction of the file or for example, double encryption whereby the code holds the other second key which is obfuscated within other code or where the code (agent) sends it on a mix path (to be decrypted at a different server and returned) if and only if the environment of the initial accessor's (key holder's) machine is whitelisted or otherwise authorized.

4) As described earlier, the probabilistic model for SDI-SCAM's distributed agents does a much better more comprehensive and (most remarkably) a faster job at detecting patterns in software, behavior, sequences, etc., than could be achieved by other state of the art means like virus scanning software. Nonetheless, the fact remains that a lot of rogue software is new, evolving, difficult to detect (e.g., dormant) or maybe "normal" software with hidden vulnerabilities readily exploitable by hackers and/or malware. Probabilistic modeling of likely software, servers and machines that may be unsafe should include the communication patterns indicating the possibility of infection by unintentional access to rogue machines, servers, software or individuals including communication with those software, servers and machines which may have come in contact with others which are suspicious. Such communication patterns (particularly changing ones) may also provide important clues of after the fact infection once a machine has been infected and taken control of e.g., from a hacker or a botnet. The other point to be made in this regard is that code (and by extension is associated machines) are particularly vulnerable to attack if the code itself is not secure. Thus, in determining probability of infection or intrusion, it is useful to apply SDI-SCAM's probabilistic modeling to the assessment of detection, likelihood and characterization of vulnerabilities of each piece of software on the network which, in turn, feeds into the broader general purpose probability analysis of intrusion and infection of software and machines across the network as herein explained.

The persistent distributed agent monitoring is particularly effective in monitoring rogue software at work in the network (or in one of SDI-SCAM's quarantined environments) and hackers being observed (in practice or in one of the system's honey pot traps) is revealing (thus can be data mined autonomously or via human in the loop) to observe and predict potential triggers (e.g. conditions, environments, data, potential targets) which certain types of suspected or identified rogue code or individuals will likely respond to so as to thus elicit their true nefarious natures or to test other triggers to learn more about them. Thus, the adaptive learning model can be learning by not just observing but also by learning triggers that certain types/characterized known "bad actors" tend respond to in a revelatory way thus providing an optimal testing strategy for triggering identifying information about it (is it unsafe or not) and secondly, what are its malicious characteristics or ways in which it can do harm. Particularly when dealing with minimal information a decision tree is a good type of decision-making tool to assess the best tests/sequence thereof for the actor based upon its own limited data profile in view of the overall probabilistic model of the system. This is because a decision tree is very efficient at spanning the space of probabilistically most likely heuristics (via associated tests) most quickly and efficiently in order from highest to lowest relevancy.

In addition to its role in providing certain key knowledge to the overall probabilistic assessment of probability and characterization of infection, intrusion and rogueness of programs, machines and individuals across the network, the application of SDI-SCAM's distributed intelligence providing a "telescopic view" of symptomatic software vulnerabilities is of course also key to ascertaining the identification and characterization of vulnerabilities (particularly the less obvious ones). It may readily tie into thus making more adaptive and comprehensive in a total data volume and data analytic sense existing rule sets for vulnerability detection software and vulnerability anti-exploitation software as well as determining the most effective solutions to recommend to remedy the identified problem, temporarily and more permanently (e.g. patches or redesign in further iterations). SDI-SCAM's distributed sensor network is particularly effective at aggregating and analyzing data across a plethora of installations of a software program, monitoring potentially a variety of heuristics indicative of potential vulnerabilities. Human in the loop or autonomous data analytic insights are thus provided into what vulnerabilities exist and what updates, patches etc. to use in future iterations of the software. Adaptive learning based upon relevance feedback from trial and error may be used to make recommendations thus providing refinements and evolve an adaptive model in order to make recommendations for example for more effective shorter-term solutions such as recommending patches and anti-exploitation solutions which are more specifically targeted to the specific vulnerabilities identified and characterized. In short, SDI-SCAM should do well in detecting, characterizing etc. vulnerabilities particularly so for applying its distributed sensor intelligence to recommending and delivering potentially a host preemptive defensive measures once it has identified and characterized existing and potential vulnerabilities including the possibility of recommending further tests to verify and/or further characterize them.

Tell-tale signs of failure of a software program, a machine or even entire computer network which is insecure could be a big data problem using perhaps non-linear kernel regression methods. Sequential progression/escalation of conditions which presage slowdown or failure are detectable even when such conditions are manifested via combinatorial heuristics. SDI-SCAM's probabilistic comprehensive assessment of vulnerabilities and comprehensive threats can feed into an alternative predictive model developed for predicting failures. In other applications of this concept, a large variety of sensors are used across a plethora of the mechanical systems (particularly engines) of commercial aircraft, fed into a database for "big data" data mining and improved prevention and diagnosis of both impending problems (on an individual basis) and general areas of vulnerability (on a general system engineering level) are all possible. In any event, such a system-level application of SDI-SCAM can make prescient predictions of "trigger" conditions or actions to avoid getting into a "yellow" or caution alert state and those to avoid to prevent getting into a red state (failure or impending failure) once in a yellow state. Or conversely, remediatory actions given any set of current conditions (which are evolving and adaptively learned by trial and error). Of course, the determination of the very fact that conditions characterizing yellow state potentially presaging red state are also part of the big data analysis. Determination/detection and best response (remedial actions) for each associated (probabilistic) alert state and associated set of conditions all seem feasible. Time sequence analytical approaches (e.g. non-linear kernel regression methods) are particularly useful in modeling, identifying responding to (or remediating) temporally evolving system states.

Previous discussions have focused principally upon determining/characterizing general system vulnerabilities (of overall value in software system (re) design and upgrading) but could be useful with regards to individual time-sensitive detection, response/remediation, problem or failure likelihood assessment as well as overall preventative operations strategies for a given software system. The same strategy of big data collection involving a variety (perhaps many) sensors across a large number of objects may be used in determining early warning detection/characterization profiles elucidating presence of conditions associated with (or presaging) potential failures for other types of complex systems like the grid (e.g. blackouts) where SDI-SCAM network of agents monitor for intrusions and infections as well as a host of other variables which are well known in the field of the particular art.

Those skilled in the art will also appreciate that the invention may be applied to other applications and may be modified without departing from the scope of the invention. Accordingly, the scope of the invention is not intended to be limited to the exemplary embodiments described above, but only by the appended claims.

What is claimed is:

1. A system that detects the state of a computer network, comprising:

a plurality of distributed agents disposed in said computer network, each said distributed agent comprising:

at least one sensor that analyzes network traffic data to passively collect, monitor, and aggregate data representative of activities of respective nodes within said computer network;

a distributed adaptive machine learning model that analyzes the aggregated data from the at least one sensor to develop activity models based on collected data and representative activities of the network in a normal state and activities of the computer network in an abnormal state as a result of at least one of intrusions, infections, scams, or suspicious activities in the computer network, wherein analysis of the aggregated data includes performing a pattern analysis on the aggregated data to identify patterns in the aggregated data representative of suspicious activities and providing adaptive rules and a probabilistic model for predicting existing threats, emerging threats, or anticipated threats to said computer network that are updated with relevance feedback, the distributed adaptive learning model further generating counteroffensive measures in response to a predicted existing threat, emerging threat, or anticipated threat to said computer network based on the relevance feedback, wherein the relevance feedback includes trial and error from previously delivered responses to previous threats and attacks on the computer network; and

means for communicating at least the aggregated data to other distributed agents on a peer-to-peer basis.

2. A system as in claim **1**, wherein the at least one sensor scans code on at least one node in the computer network for patterns that have previously been associated with viruses or malicious programming and checks incoming files to the at least one node against known viruses.

3. A system as in claim **1**, wherein the at least one sensor monitors code on the at least one node in the computer network for behavior or data traffic patterns consistent with viral infection.

4. A system as in claim **1**, wherein the at least one sensor monitors patterns of usage of at least one node in the computer network to identify patterns of usage consistent with a threat to the computer network.

5. A system as in claim **1**, wherein the at least one sensor uses natural language processing methods to analyze text for irregularities consistent with a non-human origin or to compare messages in the computer network to previously logged deceptions.

6. A system as in claim **1**, wherein the distributed adaptive learning model generates a bogus target for invoking attack on the bogus target and collects data related to the invoked attack for detecting a system infection.

7. A system as in claim **1**, further comprising a security network that is isolated from said computer network, wherein the distributed agents communicate with each other over the security network.

8. A system as in claim **1**, wherein each distributed agent shares responsibility for network traffic data analysis with at least one other distributed agent.

9. A system that detects the state of a computer network, comprising:

a plurality of distributed agents disposed in said computer network, each said distributed agent comprises at least one sensor that analyzes network traffic data to passively collect, monitor, and aggregate data representative of activities of respective nodes within said computer network and means for communicating at least the aggregated data to other distributed agents on a peer-to-peer basis; and

an adaptive machine learning model that analyzes the aggregated data from the at least one sensor to develop activity models based on collected data and representative activities of the network in a normal state and activities of the computer network in an abnormal state as a result of at least one of intrusions, infections, scams, or suspicious activities in the computer network, wherein analysis of the aggregated data includes performing a pattern analysis on the aggregated data to identify patterns in the aggregated data representative of suspicious activities and providing adaptive rules and a probabilistic model for predicting existing threats, emerging threats, or anticipated threats to said computer network that are updated with relevance feedback, the distributed adaptive learning model further generating counteroffensive measures in response to a predicted existing threat, emerging threat, or anticipated threat to said computer network based on the relevance feedback, wherein the relevance feedback includes trial and error from previously delivered responses to previous threats and attacks on the computer network.

10. A system as in claim **9**, wherein the at least one sensor scans code on at least one node in the computer network for patterns that have previously been associated with viruses or malicious programming and checks incoming files to the at least one node against known viruses.

11. A system as in claim **9**, wherein the at least one sensor monitors code on the at least one node in the computer network for behavior or data traffic patterns consistent with viral infection.

12. A system as in claim **9**, wherein the at least one sensor monitors patterns of usage of at least one node in the computer network to identify patterns of usage consistent with a threat to the computer network.

13. A system as in claim **9**, wherein the at least one sensor uses natural language processing methods to analyze text for irregularities consistent with a non-human origin or to compare messages in the computer network to previously logged deceptions.

14. A system as in claim **9**, wherein the distributed adaptive learning model generates a bogus target for invoking attack on the bogus target and collects data related to the invoked attack for detecting a system infection.

15. A system as in claim **9**, further comprising a security network that is isolated from said computer network, wherein the distributed agents communicate with each other over the security network.

16. A system as in claim **9**, wherein each distributed agent shares responsibility for network traffic data analysis with at least one other distributed agent.

* * * * *

Exhibit E

to

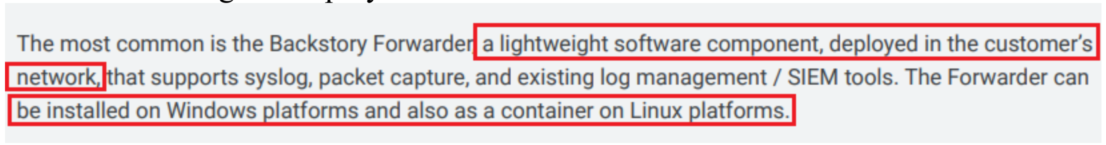
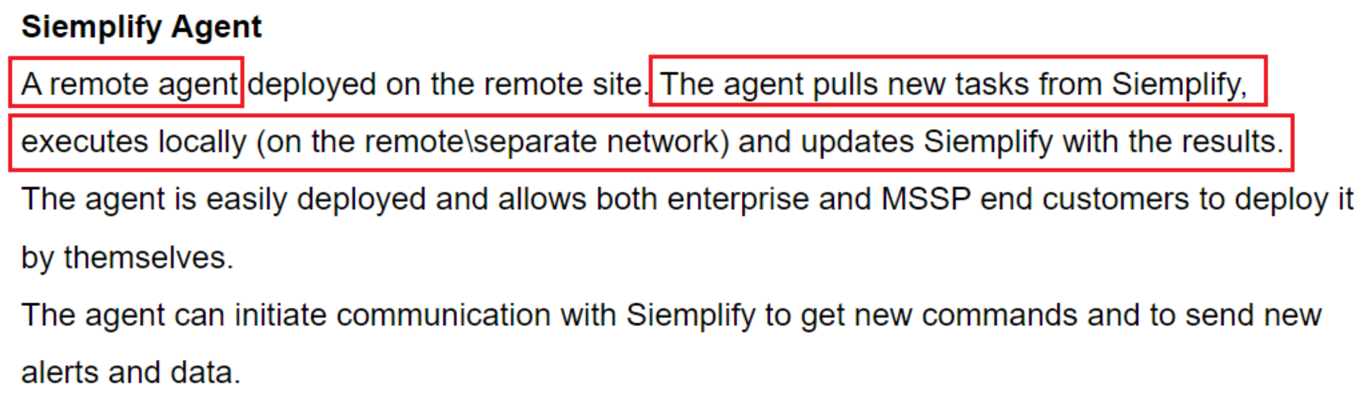
Complaint for Patent Infringement

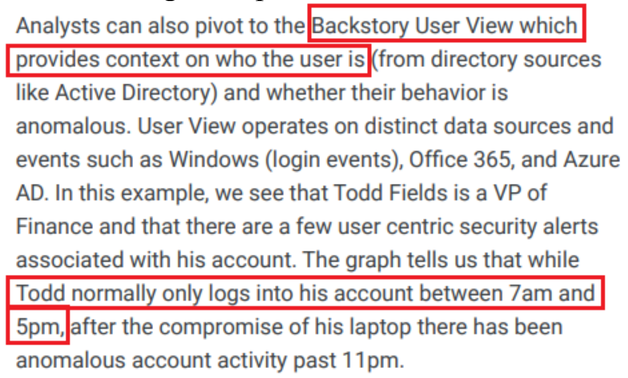
Claim Chart[1] for the ’442 Patent

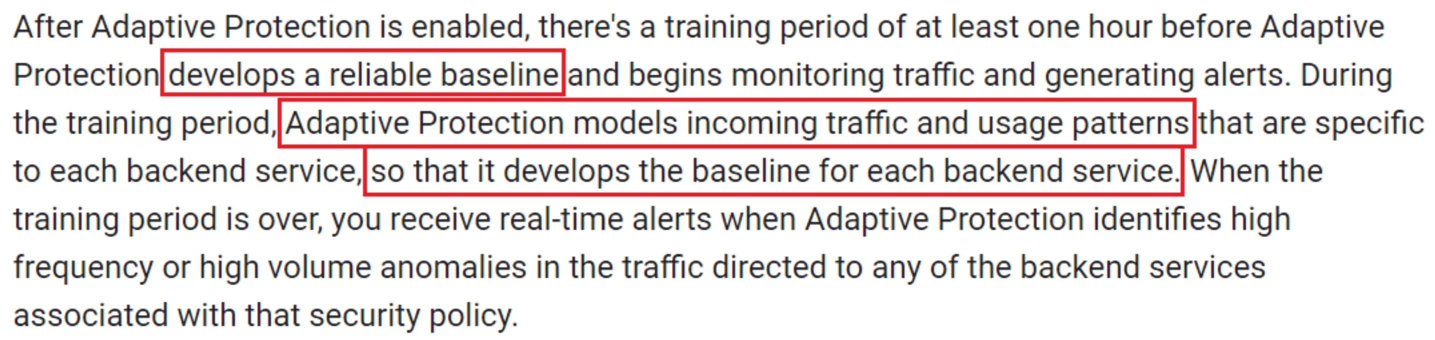
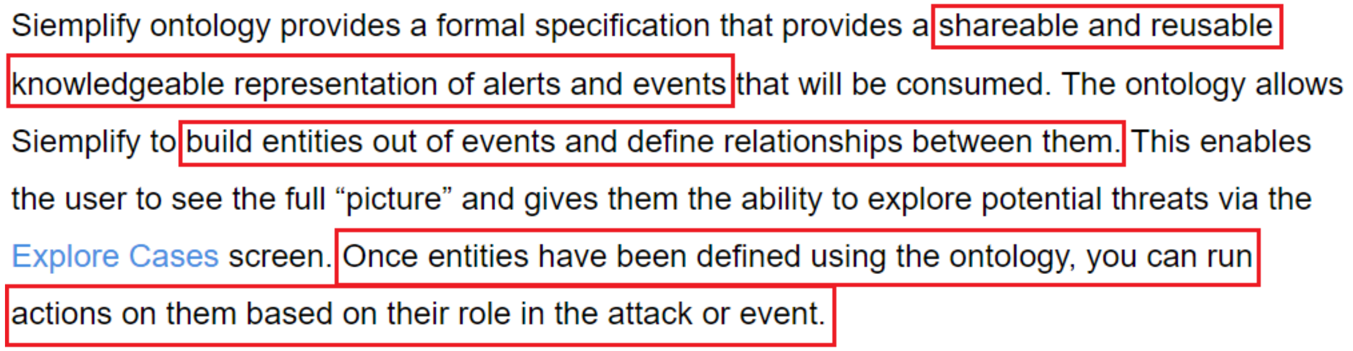
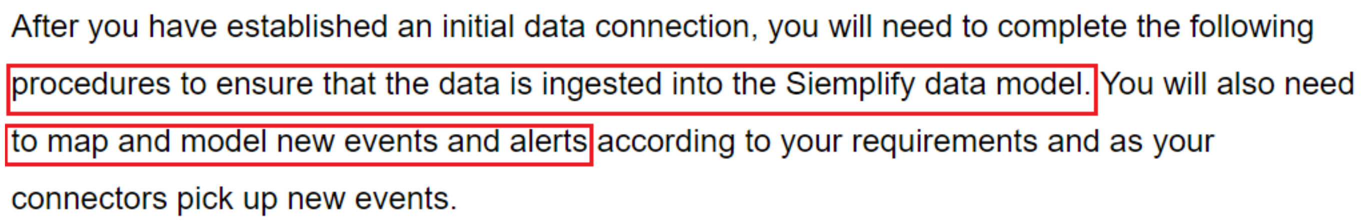
[1] Plaintiff provides this exemplary claim chart for the purposes of showing one basis of infringement of one of the Patents-in-suit by Defendant’s Accused Products as defined in the Complaint. This exemplary claim chart addresses the Accused Products broadly based on the fact that the Accused Products infringe in the same general way. Plaintiff reserves its right to amend and fully provide its infringement arguments and evidence thereof until its Preliminary and Final Infringement Contentions are later produced according to the court’s scheduling order in this case.

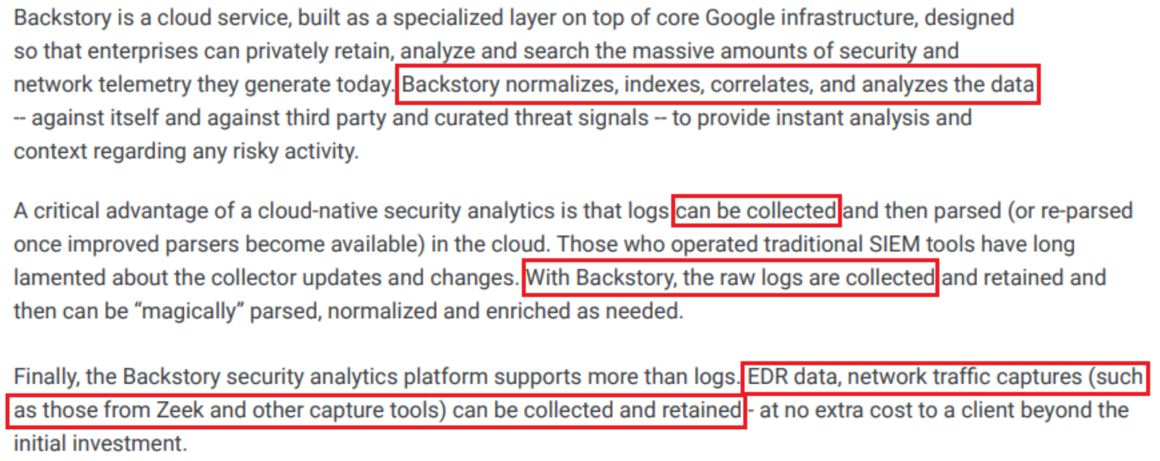
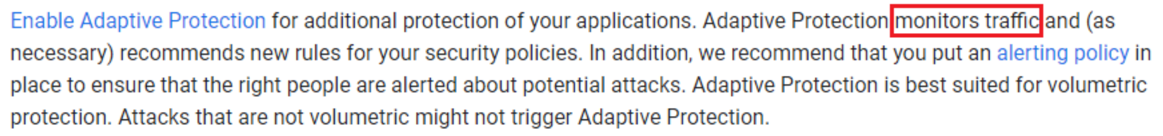
CLAIM CHART

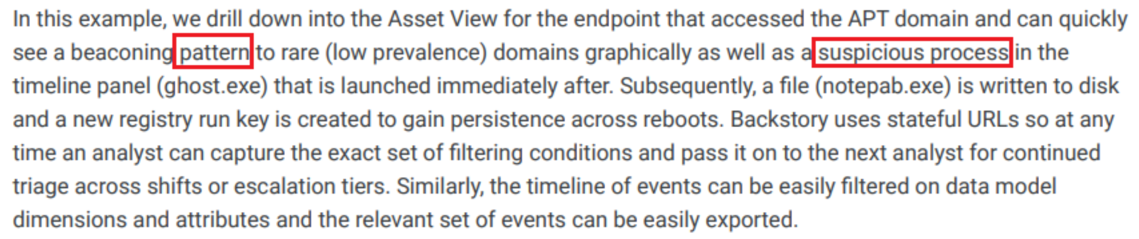
U.S. PATENT NO. 8,327,442 B2 – CLAIM 1

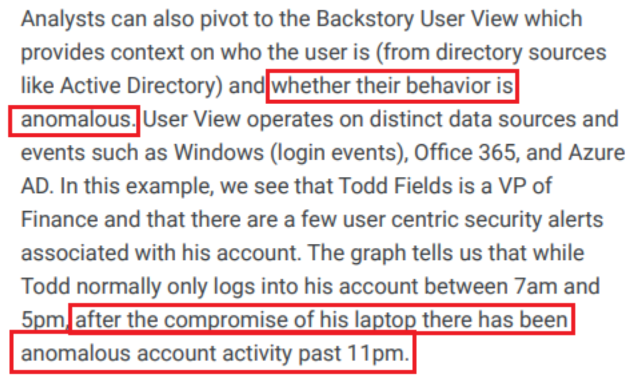
Claim 1	Corresponding Structure in Accused Systems – Google LLC
[1a] A distributed security system that protects individual computers in a computer network having a plurality of computers, said system comprising individual computers having agents associated therewith that control the associated individual computer, each agent performing the steps of:	Google LLC ("Google") provides a cloud service known as Google Cloud that contains cybersecurity services including Chronicle, Siemplify SOAR, Google Web Risk, and Google Cloud Armor. *See* https://cloud.google.com/products, https://chronicle.security/product/, https://cloud.google.com/solutions/security-orchestration-automation-response, https://cloud.google.com/web-risk, and https://cloud.google.com/armor. Google's Backstory service can use the Backstory Forwarder as an agent to collect network data. Siemplify SOAR can use remote agents deployed on customer networks.  The most common is the Backstory Forwarder a lightweight software component, deployed in the customer's network, that supports syslog, packet capture, and existing log management / SIEM tools. The Forwarder can be installed on Windows platforms and also as a container on Linux platforms. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)  **Siemplify Agent** A remote agent deployed on the remote site. The agent pulls new tasks from Siemplify, executes locally (on the remote\separate network) and updates Siemplify with the results. The agent is easily deployed and allows both enterprise and MSSP end customers to deploy it by themselves. The agent can initiate communication with Siemplify to get new commands and to send new alerts and data. **Source:** (https://documents.siemplify.co/en/articles/4-remote-agents-for-cloud-overview) Google Cloud together with various equipment, services, components, and/or software utilized in providing Google Cloud collectively include a system and method for a distributed application and network security system (SDI-SCAM) as described by the meaning of this claim. Google Cloud is made available by a system owned and/or operated by Google.

	"Because infringement liability is not dependent on ownership, e.g., use of a system can infringe (35 U.S.C. § 271), infringement is not dependent on ownership of all limitations of a claim."
[1b] creating statistical models of usage of the associated individual computer in said computer network;	Google's Backstory service can determine normal usage behavior and provide context through Backstory User View. Google Cloud Armor can use Adaptive Protection which establishes a baseline model of network traffic and usage patterns. Siemplify SOAR can model new events and alerts into shareable and reusable entities through ontology. The following exemplifies this limitation's existence in Accused Systems:  Analysts can also pivot to the Backstory User View which provides context on who the user is (from directory sources like Active Directory) and whether their behavior is anomalous. User View operates on distinct data sources and events such as Windows (login events), Office 365, and Azure AD. In this example, we see that Todd Fields is a VP of Finance and that there are a few user centric security alerts associated with his account. The graph tells us that while Todd normally only logs into his account between 7am and 5pm, after the compromise of his laptop there has been anomalous account activity past 11pm. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)

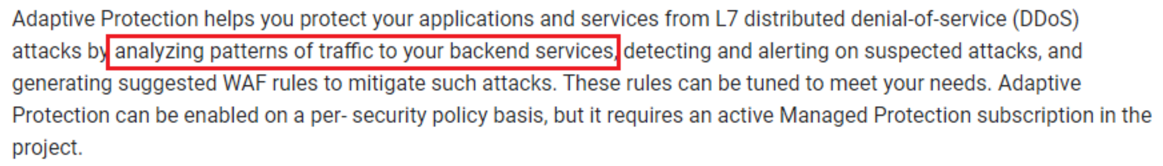
	 After Adaptive Protection is enabled, there's a training period of at least one hour before Adaptive Protection develops a reliable baseline and begins monitoring traffic and generating alerts. During the training period, Adaptive Protection models incoming traffic and usage patterns that are specific to each backend service, so that it develops the baseline for each backend service. When the training period is over, you receive real-time alerts when Adaptive Protection identifies high frequency or high volume anomalies in the traffic directed to any of the backend services associated with that security policy. **Source:** (https://cloud.google.com/armor/docs/adaptive-protection-overview)  Siemplify ontology provides a formal specification that provides a shareable and reusable knowledgeable representation of alerts and events that will be consumed. The ontology allows Siemplify to build entities out of events and define relationships between them. This enables the user to see the full "picture" and gives them the ability to explore potential threats via the Explore Cases screen. Once entities have been defined using the ontology, you can run actions on them based on their role in the attack or event.  After you have established an initial data connection, you will need to complete the following procedures to ensure that the data is ingested into the Siemplify data model. You will also need to map and model new events and alerts according to your requirements and as your connectors pick up new events. **Source:** (https://documents.siemplify.co/en/articles/454-ontology-overview)

[1c] gathering and analyzing information relating to current usage of the associated individual computer in said computer network;	Google's Backstory service can collect, correlate, and analyze telemetry data and logs. Google Cloud Armor can use Adaptive Protection to monitor traffic. The following exemplifies this limitation's existence in Accused Systems:  Backstory is a cloud service, built as a specialized layer on top of core Google infrastructure, designed so that enterprises can privately retain, analyze and search the massive amounts of security and network telemetry they generate today. Backstory normalizes, indexes, correlates, and analyzes the data -- against itself and against third party and curated threat signals -- to provide instant analysis and context regarding any risky activity. A critical advantage of a cloud-native security analytics is that logs can be collected and then parsed (or re-parsed once improved parsers become available) in the cloud. Those who operated traditional SIEM tools have long lamented about the collector updates and changes. With Backstory, the raw logs are collected and retained and then can be "magically" parsed, normalized and enriched as needed. Finally, the Backstory security analytics platform supports more than logs. EDR data, network traffic captures (such as those from Zeek and other capture tools) can be collected and retained - at no extra cost to a client beyond the initial investment. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)  Enable Adaptive Protection for additional protection of your applications. Adaptive Protection monitors traffic and (as necessary) recommends new rules for your security policies. In addition, we recommend that you put an alerting policy in place to ensure that the right people are alerted about potential attacks. Adaptive Protection is best suited for volumetric protection. Attacks that are not volumetric might not trigger Adaptive Protection. **Source:** (https://cloud.google.com/armor/docs/best-practices)
[1d] determining from said information a pattern of usage of the associated individual computer that is consistent with intrusion or attack of the associated individual computer or the computer network;	Google's Backstory service can identify suspicious behavioral patterns. Google Cloud Armor can use Adaptive Protection to analyze network traffic patterns for suspicious activity and create models to detect their signatures. Google Chronicle's VirusTotal service can process and extract patterns from data. The following exemplifies this limitation's existence in Accused Systems:

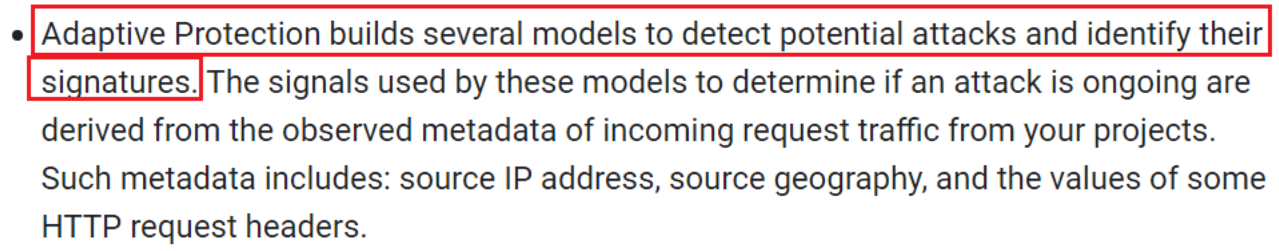

In this example, we drill down into the Asset View for the endpoint that accessed the APT domain and can quickly see a beaconing pattern to rare (low prevalence) domains graphically as well as a suspicious process in the timeline panel (ghost.exe) that is launched immediately after. Subsequently, a file (notepab.exe) is written to disk and a new registry run key is created to gain persistence across reboots. Backstory uses stateful URLs so at any time an analyst can capture the exact set of filtering conditions and pass it on to the next analyst for continued triage across shifts or escalation tiers. Similarly, the timeline of events can be easily filtered on data model dimensions and attributes and the relevant set of events can be easily exported.


Analysts can also pivot to the Backstory User View which provides context on who the user is (from directory sources like Active Directory) and whether their behavior is anomalous. User View operates on distinct data sources and events such as Windows (login events), Office 365, and Azure AD. In this example, we see that Todd Fields is a VP of Finance and that there are a few user centric security alerts associated with his account. The graph tells us that while Todd normally only logs into his account between 7am and 5pm, after the compromise of his laptop there has been anomalous account activity past 11pm.

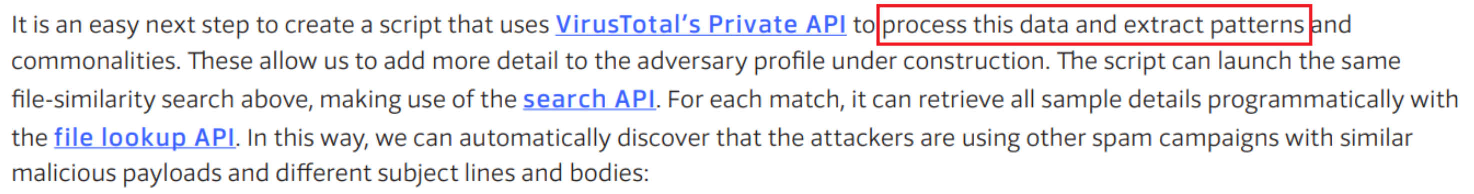
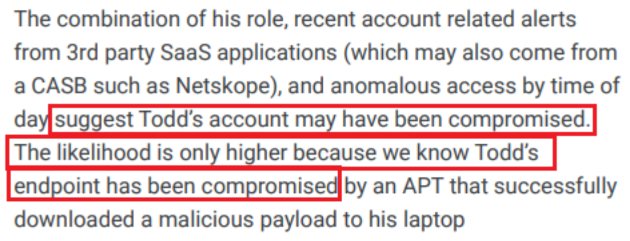
Source: (https://go.chronicle.security/hubfs/Backstory_WP.pdf)

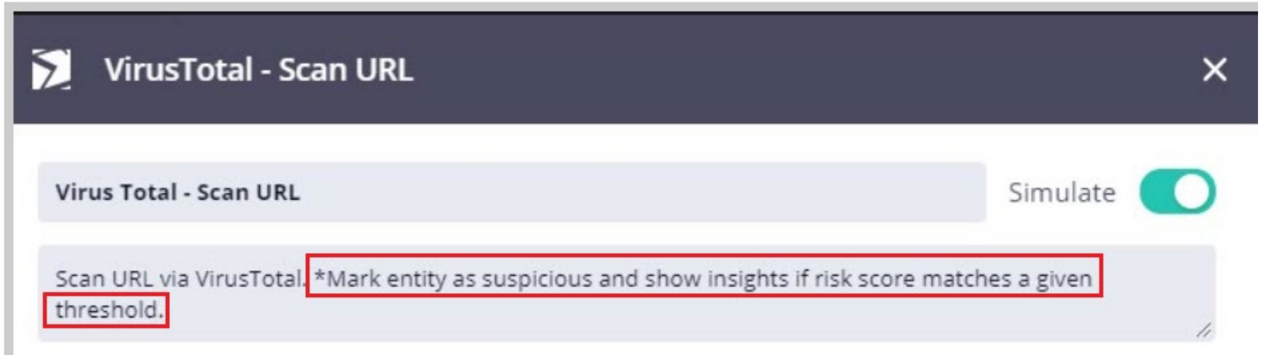

Adaptive Protection helps you protect your applications and services from L7 distributed denial-of-service (DDoS) attacks by analyzing patterns of traffic to your backend services, detecting and alerting on suspected attacks, and generating suggested WAF rules to mitigate such attacks. These rules can be tuned to meet your needs. Adaptive Protection can be enabled on a per- security policy basis, but it requires an active Managed Protection subscription in the project.

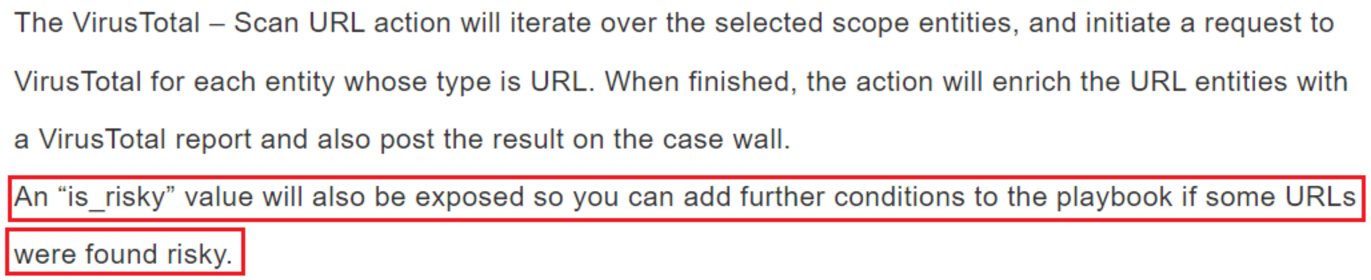
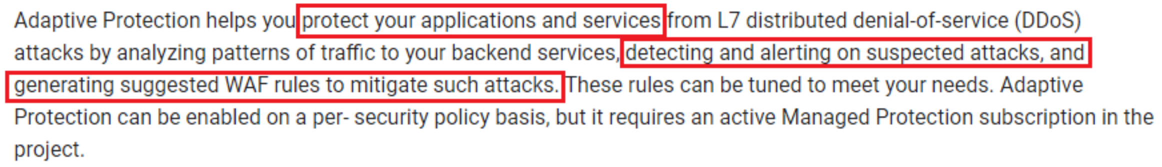
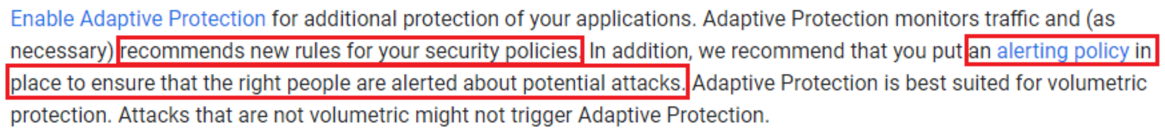
Source: (https://cloud.google.com/armor/docs/cloud-armor-overview)

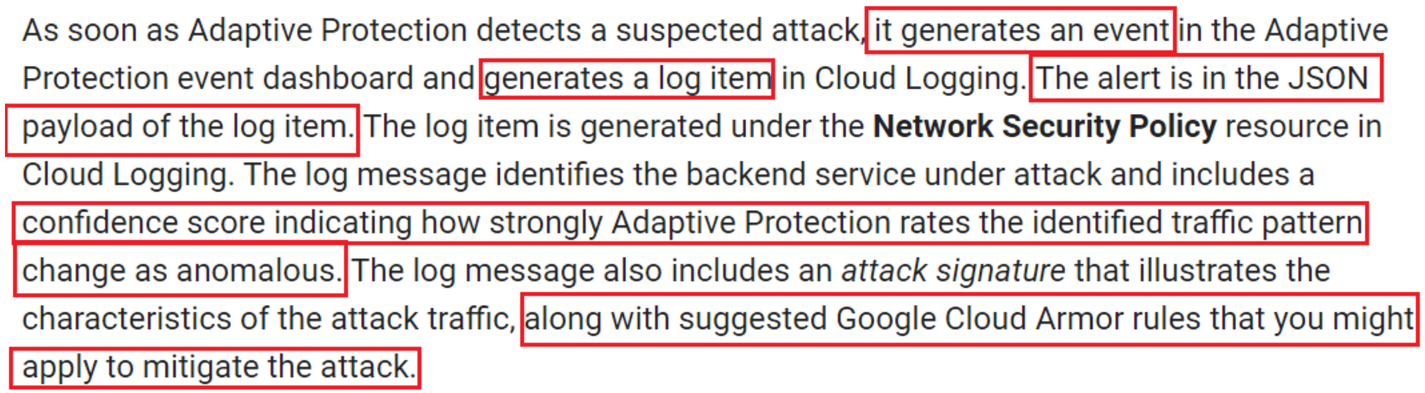
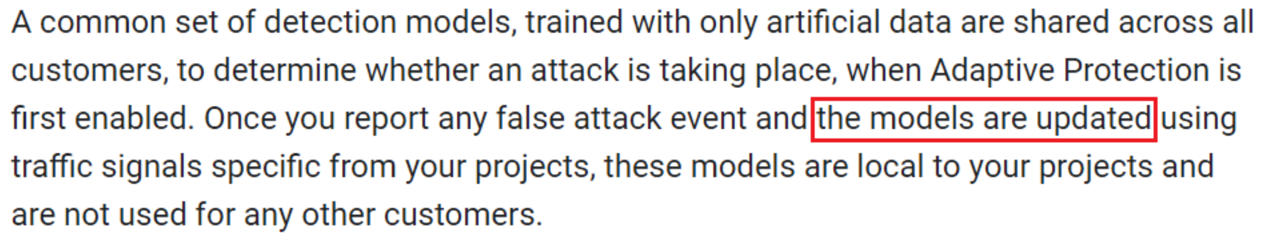
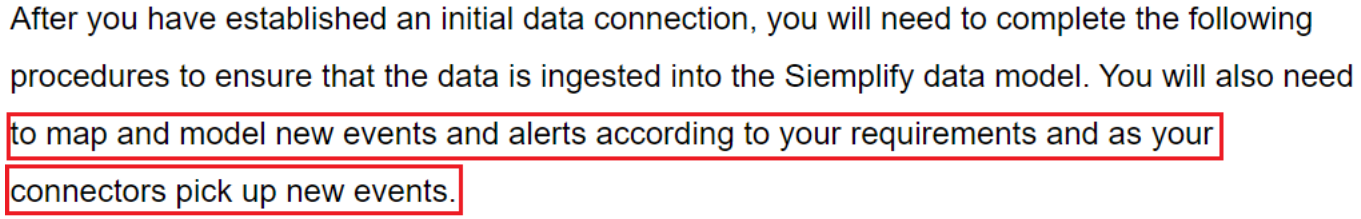

- Adaptive Protection builds several models to detect potential attacks and identify their signatures. The signals used by these models to determine if an attack is ongoing are derived from the observed metadata of incoming request traffic from your projects. Such metadata includes: source IP address, source geography, and the values of some HTTP request headers.

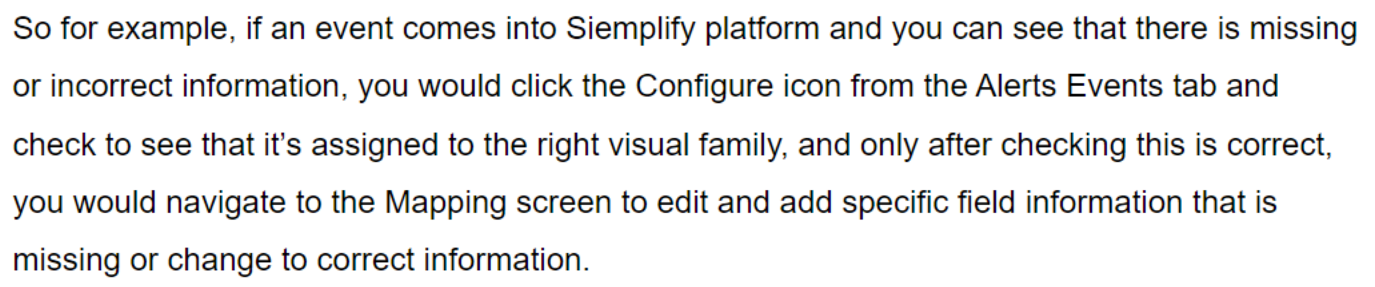
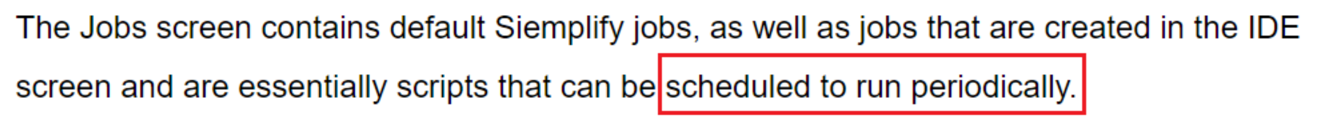
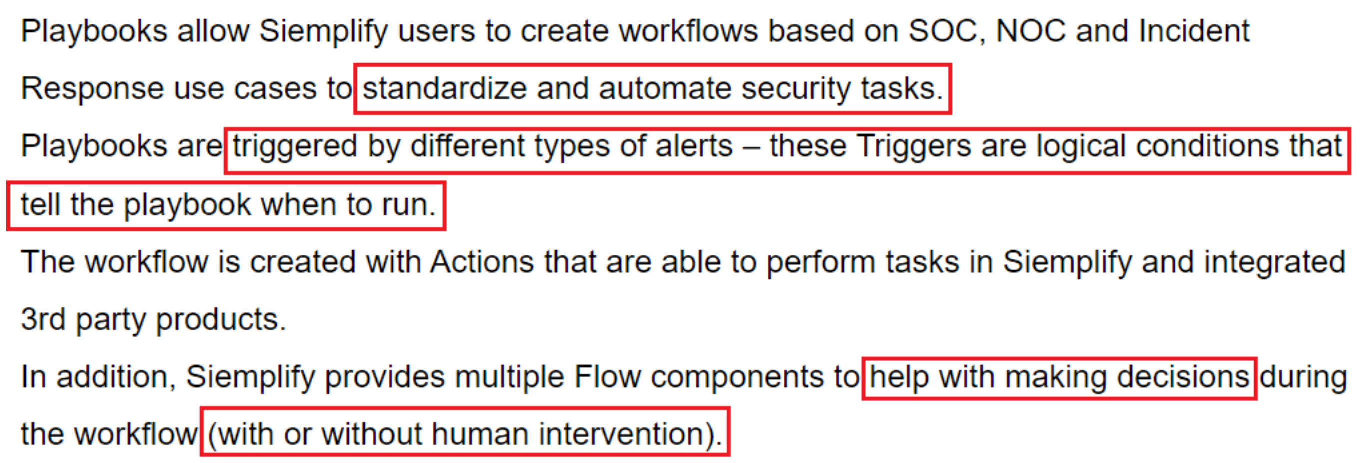
Source: (https://cloud.google.com/armor/docs/adaptive-protection-overview)

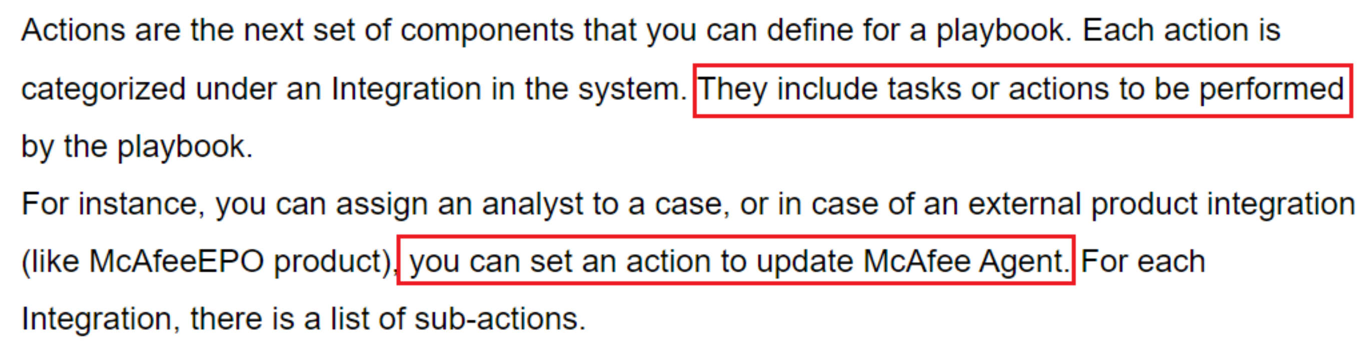
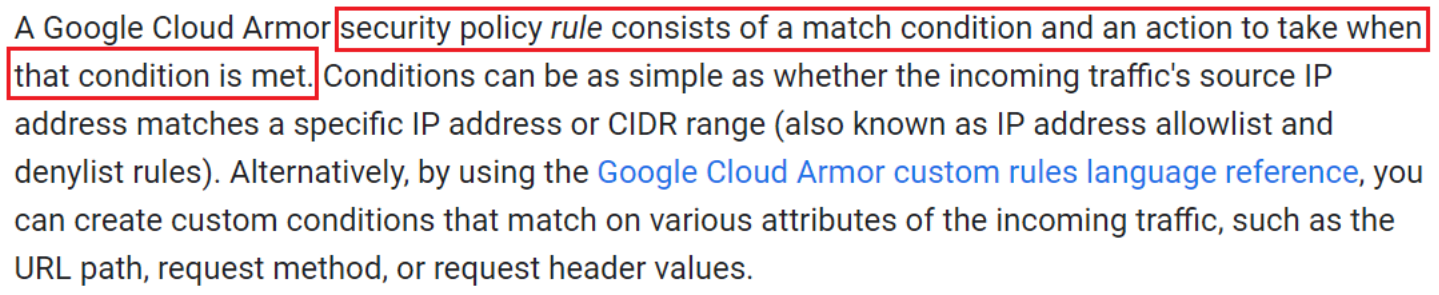
	 It is an easy next step to create a script that uses VirusTotal's Private API to process this data and extract patterns and commonalities. These allow us to add more detail to the adversary profile under construction. The script can launch the same file-similarity search above, making use of the search API. For each match, it can retrieve all sample details programmatically with the file lookup API. In this way, we can automatically discover that the attackers are using other spam campaigns with similar malicious payloads and different subject lines and bodies: **Source:** (https://go.chronicle.security/hubfs/VirusTotal_Enterprise_WP.pdf)
[1e] determining a probability of the likelihood of an intrusion or attack from said pattern of usage of the associated individual computer;	Google's Backstory service can provide insight on the probability of a compromised device. Google Cloud Armor can provide a confidence level for the likelihood that a change is anomalous. Siemplify SOAR can use VirusTotal to scan URLs and mark them as suspicious if their risk score matches a certain threshold. The following exemplifies this limitation's existence in Accused Systems:  The combination of his role, recent account related alerts from 3rd party SaaS applications (which may also come from a CASB such as Netskope), and anomalous access by time of day suggest Todd's account may have been compromised. The likelihood is only higher because we know Todd's endpoint has been compromised by an APT that successfully downloaded a malicious payload to his laptop **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)

	You can tune Adaptive Protection alerts based on several metrics. The alerts, which are sent to Cloud Logging, include a *confidence level*, an *attack signature*, a suggested rule, and an estimated *impacted baseline rate* associated with the suggested rule. • The confidence level indicates the confidence with which Adaptive Protection models predict that the observed change in traffic pattern is anomalous. • The impacted baseline rates associated with the suggested rule represent the percentage of existing baseline traffic that is caught by the rule. Two rates are provided. The first is the percentage relative to the traffic into the specific backend service under attack. The second is the percentage relative to all the traffic going through the security policy including all configured backend service targets (not just the one being attacked). **Source:** (https://cloud.google.com/armor/docs/adaptive-protection-overview)  VirusTotal - Scan URL Virus Total - Scan URL Simulate Scan URL via VirusTotal. *Mark entity as suspicious and show insights if risk score matches a given threshold. **Source:** (https://documents.siemplify.co/en/articles/417-working-with-playbook-simulator)

	 The VirusTotal – Scan URL action will iterate over the selected scope entities, and initiate a request to VirusTotal for each entity whose type is URL. When finished, the action will enrich the URL entities with a VirusTotal report and also post the result on the case wall. An "is_risky" value will also be exposed so you can add further conditions to the playbook if some URLs were found risky. **Source:** (https://documents.siemplify.co/en/articles/481-how-do-i-scan-multiple-urls-in-virustotal)
[1f] distributing in real-time warnings and potential countermeasures to agents of each of said individual computers in said computer network when the determined probability of the likelihood of an intrusion or attack exceeds a statistical threshold, wherein at least one of said warnings comprises information related to the nature of the intrusion or attack and the determined probability of the likelihood of intrusion or attack based on the statistical models of the associated individual computer; and	Google Cloud Armor can use Adaptive Protection to detect and alert on suspected attacks. It can generate events with a confidence score indicating the likelihood of the pattern change as anomalous and provide potential mitigation methods. The following exemplifies this limitation's existence in Accused Systems:  Adaptive Protection helps you protect your applications and services from L7 distributed denial-of-service (DDoS) attacks by analyzing patterns of traffic to your backend services, detecting and alerting on suspected attacks, and generating suggested WAF rules to mitigate such attacks. These rules can be tuned to meet your needs. Adaptive Protection can be enabled on a per- security policy basis, but it requires an active Managed Protection subscription in the project. **Source:** (https://cloud.google.com/armor/docs/cloud-armor-overview)  Enable Adaptive Protection for additional protection of your applications. Adaptive Protection monitors traffic and (as necessary) recommends new rules for your security policies. In addition, we recommend that you put an alerting policy in place to ensure that the right people are alerted about potential attacks. Adaptive Protection is best suited for volumetric protection. Attacks that are not volumetric might not trigger Adaptive Protection. **Source:** (https://cloud.google.com/armor/docs/best-practices)

	 As soon as Adaptive Protection detects a suspected attack, it generates an event in the Adaptive Protection event dashboard and generates a log item in Cloud Logging. The alert is in the JSON payload of the log item. The log item is generated under the **Network Security Policy** resource in Cloud Logging. The log message identifies the backend service under attack and includes a confidence score indicating how strongly Adaptive Protection rates the identified traffic pattern change as anomalous. The log message also includes an *attack signature* that illustrates the characteristics of the attack traffic, along with suggested Google Cloud Armor rules that you might apply to mitigate the attack. **Source:** (https://cloud.google.com/armor/docs/adaptive-protection-overview)
[1g] updating said statistical models of the associated individual computer to reflect the current usage of the associated individual computer in said computer network and the likelihood of intrusion or attack;	Google Cloud Armor can update models based on customers' specific traffic signals. Siemplify SOAR can model new events with ontology and correct or update information for the models. The following exemplifies this limitation's existence in Accused Systems:  A common set of detection models, trained with only artificial data are shared across all customers, to determine whether an attack is taking place, when Adaptive Protection is first enabled. Once you report any false attack event and the models are updated using traffic signals specific from your projects, these models are local to your projects and are not used for any other customers. **Source:** (https://cloud.google.com/armor/docs/adaptive-protection-overview)  After you have established an initial data connection, you will need to complete the following procedures to ensure that the data is ingested into the Siemplify data model. You will also need to map and model new events and alerts according to your requirements and as your connectors pick up new events. **Source:** (https://documents.siemplify.co/en/articles/454-ontology-overview)

	 So for example, if an event comes into Siemplify platform and you can see that there is missing or incorrect information, you would click the Configure icon from the Alerts Events tab and check to see that it's assigned to the right visual family, and only after checking this is correct, you would navigate to the Mapping screen to edit and add specific field information that is missing or change to correct information. **Source:** (https://documents.siemplify.co/en/articles/459-configure-mapping-and-assign-visual-families)
wherein each said agent schedules the associated individual computers for different anti-viral software updates based on different levels of probability of an intrusion or attack for each individual computer based on the statistical model for each individual computer and a detected level of probability of an intrusion or attack; and	Siemplify SOAR can schedule scripts to run periodically, which can include software updates. It can also use playbooks to automate security tasks, including updating anti-virus and other security software. The following exemplifies this limitation's existence in Accused Systems:  The Jobs screen contains default Siemplify jobs, as well as jobs that are created in the IDE screen and are essentially scripts that can be scheduled to run periodically. **Source:** (https://documents.siemplify.co/en/articles/512-writing-jobs)  Playbooks allow Siemplify users to create workflows based on SOC, NOC and Incident Response use cases to standardize and automate security tasks. Playbooks are triggered by different types of alerts – these Triggers are logical conditions that tell the playbook when to run. The workflow is created with Actions that are able to perform tasks in Siemplify and integrated 3rd party products. In addition, Siemplify provides multiple Flow components to help with making decisions during the workflow (with or without human intervention). **Source:** (https://documents.siemplify.co/en/articles/183-create-and-run-a-playbook)

	 Actions are the next set of components that you can define for a playbook. Each action is categorized under an Integration in the system. They include tasks or actions to be performed by the playbook. For instance, you can assign an analyst to a case, or in case of an external product integration (like McAfeeEPO product), you can set an action to update McAfee Agent. For each Integration, there is a list of sub-actions. **Source:** (https://documents.siemplify.co/en/articles/440-using-actions-in-playbooks)
wherein each said agent suspends said schedule and immediately provides the anti-viral software update to the associated individual computer when an intrusion or attack of any computer in said computer network is detected or the detected probability of an intrusion or attack is high that the associated individual computer has been infected by a particular type of virus.	Google Cloud Armor can perform actions automatically in response to a condition being met. Siemplify SOAR can use playbooks to automatically update anti-viral and security software upon being triggered by specific alerts. The following exemplifies this limitation's existence in Accused Systems:  A Google Cloud Armor security policy *rule* consists of a match condition and an action to take when that condition is met. Conditions can be as simple as whether the incoming traffic's source IP address matches a specific IP address or CIDR range (also known as IP address allowlist and denylist rules). Alternatively, by using the Google Cloud Armor custom rules language reference, you can create custom conditions that match on various attributes of the incoming traffic, such as the URL path, request method, or request header values. **Source:** (https://cloud.google.com/armor/docs/security-policy-overview)

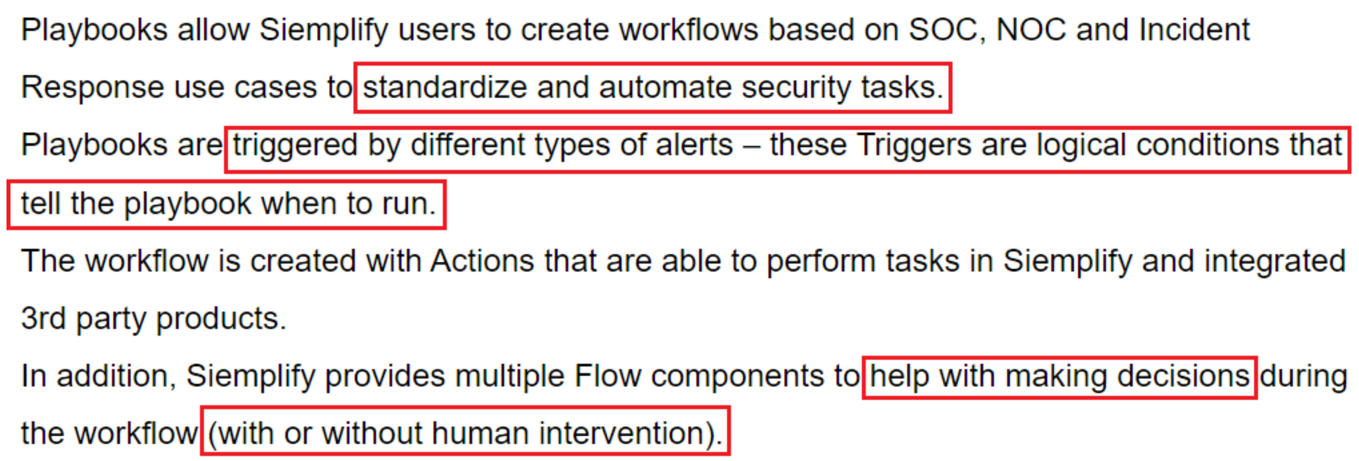
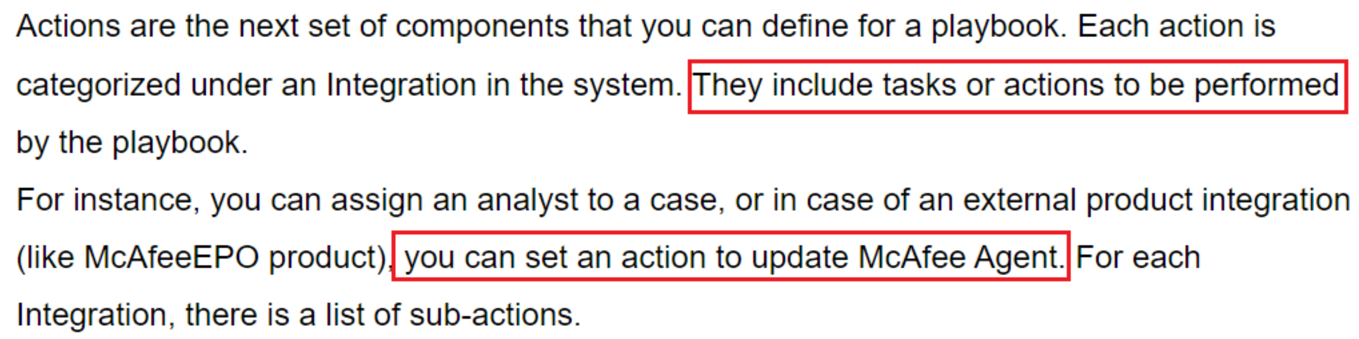
	 Playbooks allow Siemplify users to create workflows based on SOC, NOC and Incident Response use cases to standardize and automate security tasks. Playbooks are triggered by different types of alerts – these Triggers are logical conditions that tell the playbook when to run. The workflow is created with Actions that are able to perform tasks in Siemplify and integrated 3rd party products. In addition, Siemplify provides multiple Flow components to help with making decisions during the workflow (with or without human intervention). **Source:** (https://documents.siemplify.co/en/articles/183-create-and-run-a-playbook)  Actions are the next set of components that you can define for a playbook. Each action is categorized under an Integration in the system. They include tasks or actions to be performed by the playbook. For instance, you can assign an analyst to a case, or in case of an external product integration (like McAfeeEPO product), you can set an action to update McAfee Agent. For each Integration, there is a list of sub-actions. **Source:** (https://documents.siemplify.co/en/articles/440-using-actions-in-playbooks)

Exhibit F

to

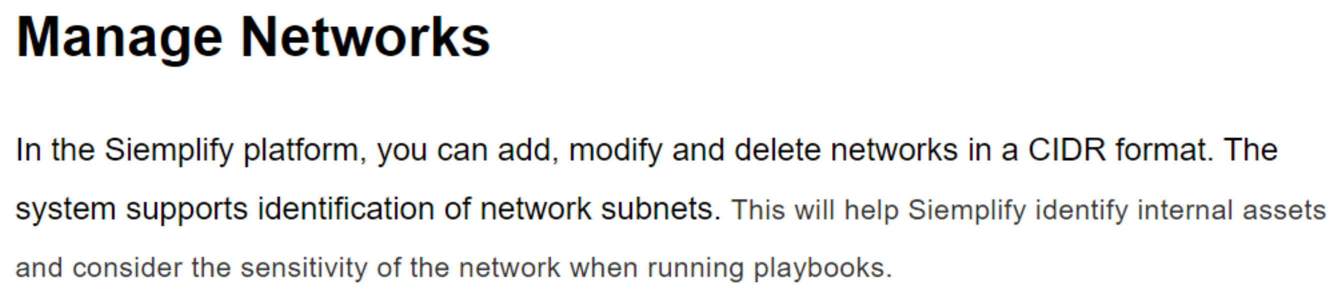
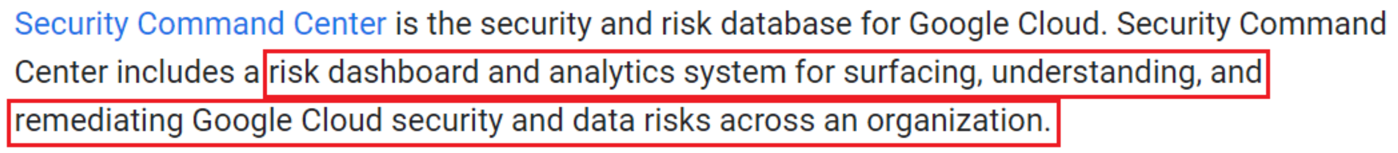
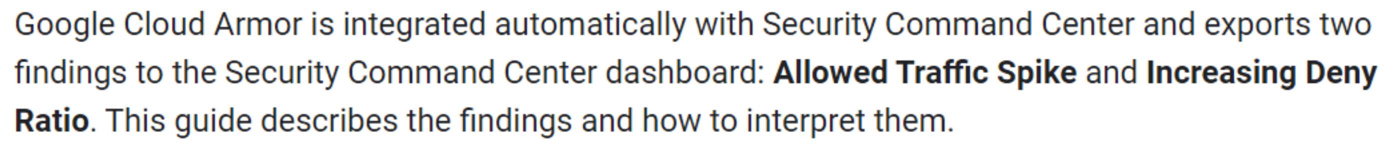
Complaint for Patent Infringement

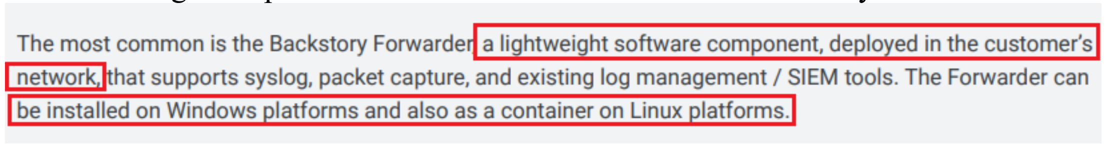
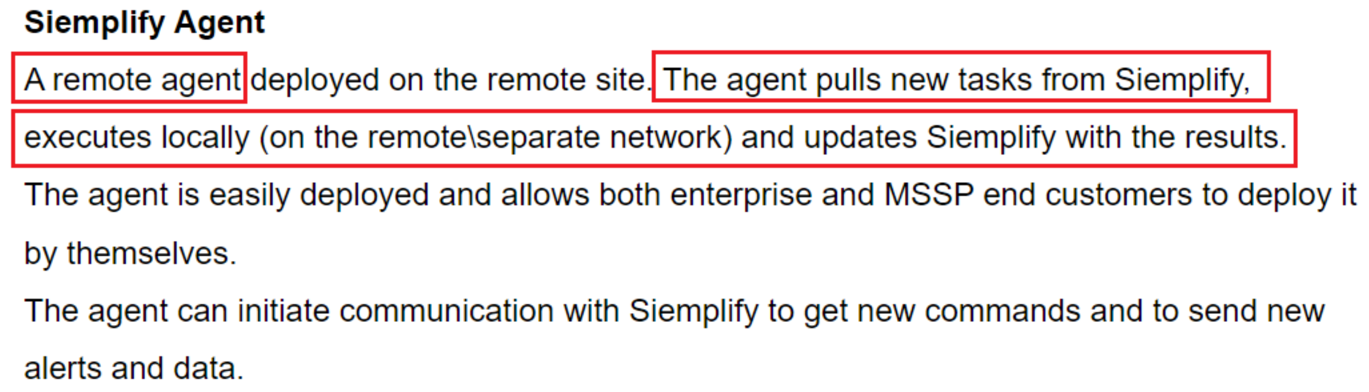
Claim Chart[1] for the ’614 Patent

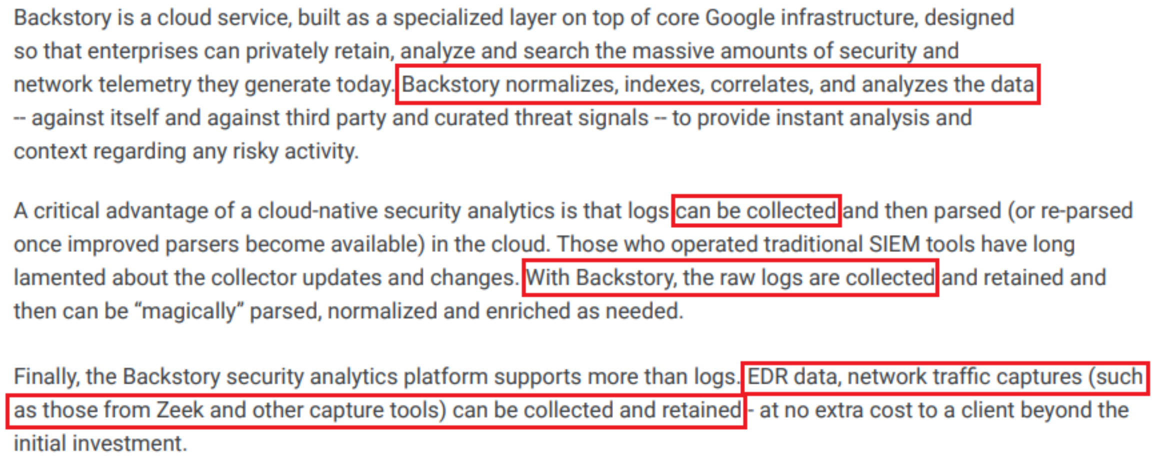
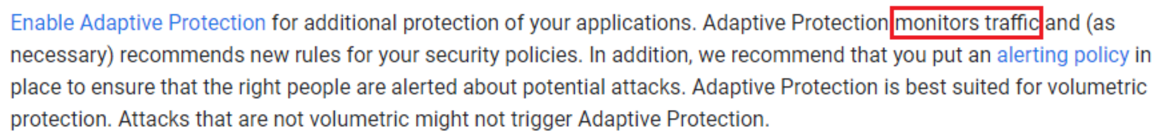
[1] Plaintiff provides this exemplary claim chart for the purposes of showing one basis of infringement of one of the Patents-in-suit by Defendant’s Accused Products as defined in the Complaint. This exemplary claim chart addresses the Accused Products broadly based on the fact that the Accused Products infringe in the same general way. Plaintiff reserves its right to amend and fully provide its infringement arguments and evidence thereof until its Preliminary and Final Infringement Contentions are later produced according to the court’s scheduling order in this case.

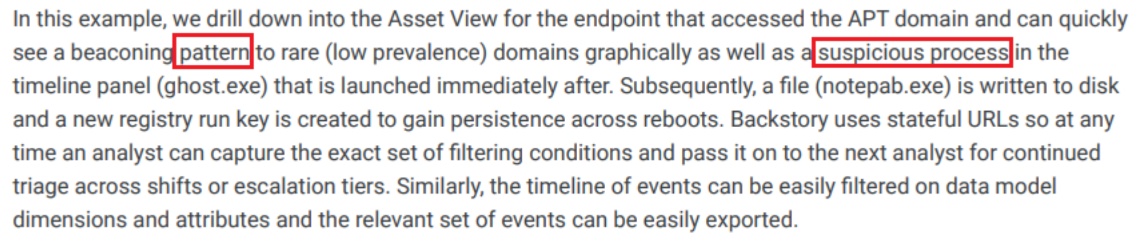
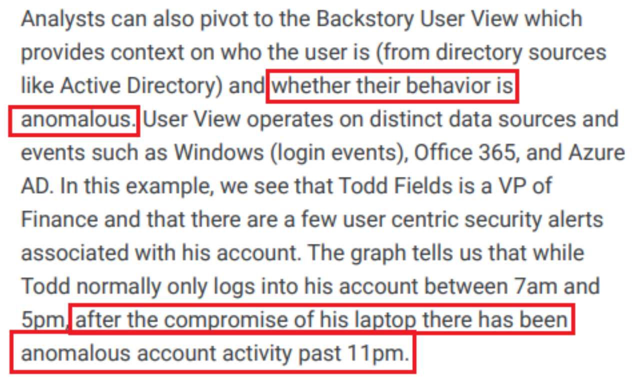
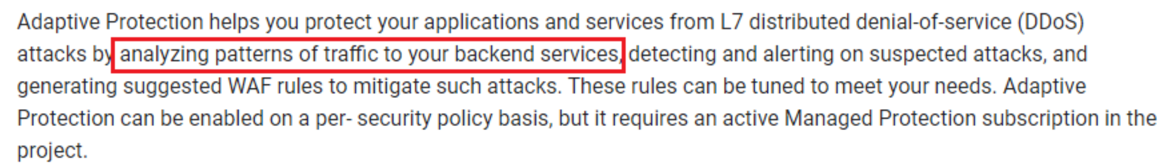
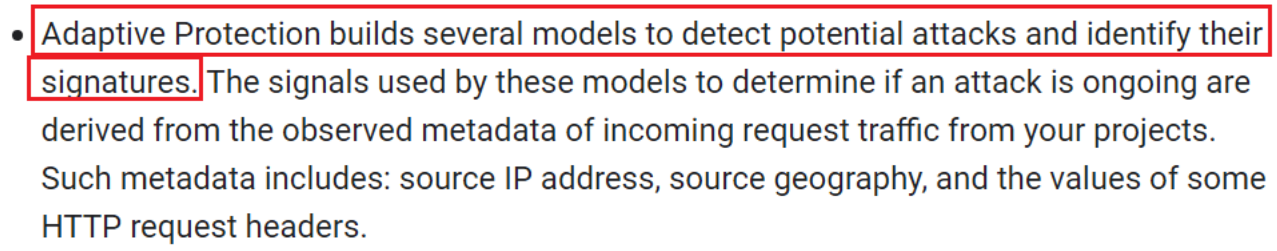
CLAIM CHART

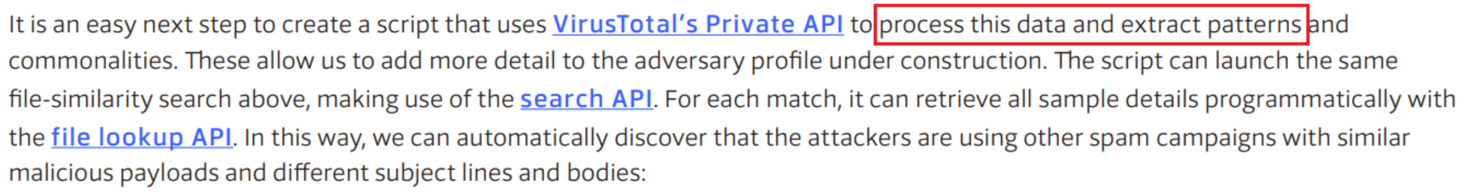
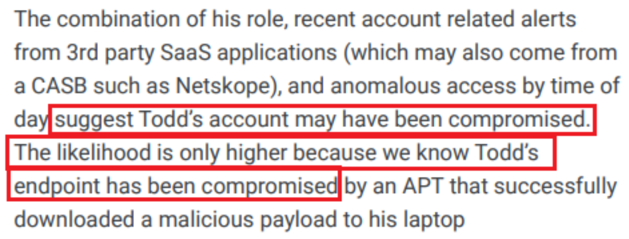
U.S. PATENT NO. 9,438,614 B2 – CLAIM 10

Claim 10	Corresponding Structure in Accused Systems – Google LLC
[10a] A system that detects the state of a computer network having a plurality of nodes,	Google LLC ("Google") provides a cloud service known as Google Cloud that contains cybersecurity services including Chronicle, Siemplify SOAR, Google Web Risk, and Google Cloud Armor. *See* https://cloud.google.com/products, https://chronicle.security/product/, https://cloud.google.com/solutions/security-orchestration-automation-response, https://cloud.google.com/web-risk, and https://cloud.google.com/armor. Siemplify SOAR can allow users to manage networks and identify subnets. Google Cloud Armor can integrate with Google's Security Command Center to view security and data risks across the organization.  **Manage Networks** In the Siemplify platform, you can add, modify and delete networks in a CIDR format. The system supports identification of network subnets. This will help Siemplify identify internal assets and consider the sensitivity of the network when running playbooks. **Source:** (https://documents.siemplify.co/en/articles/451-manage-networks)  Security Command Center is the security and risk database for Google Cloud. Security Command Center includes a risk dashboard and analytics system for surfacing, understanding, and remediating Google Cloud security and data risks across an organization.  Google Cloud Armor is integrated automatically with Security Command Center and exports two findings to the Security Command Center dashboard: **Allowed Traffic Spike** and **Increasing Deny Ratio**. This guide describes the findings and how to interpret them. **Source:** (https://cloud.google.com/armor/docs/cscc-findings)

	Google Cloud together with various equipment, services, components, and/or software utilized in providing Google Cloud collectively include a system and method for a Sdi-scam as described by the meaning of this claim. Google Cloud is made available by a system owned and/or operated by Google. "Because infringement liability is not dependent on ownership, e.g., use of a system can infringe (35 U.S.C. § 271), infringement is not dependent on ownership of all limitations of a claim."
[10b] said system comprising a plurality of distributed agents designed for adaptive learning and probabilistic analysis,	Google's Backstory service can use the Backstory Forwarder as an agent to collect network data. Siemplify SOAR can use remote agents deployed on customer networks. The following exemplifies this limitation's existence in Accused Systems: The most common is the Backstory Forwarder a lightweight software component, deployed in the customer's network, that supports syslog, packet capture, and existing log management / SIEM tools. The Forwarder can be installed on Windows platforms and also as a container on Linux platforms. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf) **Siemplify Agent** A remote agent deployed on the remote site. The agent pulls new tasks from Siemplify, executes locally (on the remote\separate network) and updates Siemplify with the results. The agent is easily deployed and allows both enterprise and MSSP end customers to deploy it by themselves. The agent can initiate communication with Siemplify to get new commands and to send new alerts and data. **Source:** (https://documents.siemplify.co/en/articles/4-remote-agents-for-cloud-overview)

[10c] said agents passively collecting, monitoring, aggregating and	Google's Backstory service can collect, correlate, and analyze telemetry data and logs. Google Cloud Armor can use Adaptive Protection to monitor traffic. The following exemplifies this limitation's existence in Accused Systems:  Backstory is a cloud service, built as a specialized layer on top of core Google infrastructure, designed so that enterprises can privately retain, analyze and search the massive amounts of security and network telemetry they generate today. Backstory normalizes, indexes, correlates, and analyzes the data -- against itself and against third party and curated threat signals -- to provide instant analysis and context regarding any risky activity. A critical advantage of a cloud-native security analytics is that logs can be collected and then parsed (or re-parsed once improved parsers become available) in the cloud. Those who operated traditional SIEM tools have long lamented about the collector updates and changes. With Backstory, the raw logs are collected and retained and then can be "magically" parsed, normalized and enriched as needed. Finally, the Backstory security analytics platform supports more than logs. EDR data, network traffic captures (such as those from Zeek and other capture tools) can be collected and retained - at no extra cost to a client beyond the initial investment. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)  Enable Adaptive Protection for additional protection of your applications. Adaptive Protection monitors traffic and (as necessary) recommends new rules for your security policies. In addition, we recommend that you put an alerting policy in place to ensure that the right people are alerted about potential attacks. Adaptive Protection is best suited for volumetric protection. Attacks that are not volumetric might not trigger Adaptive Protection. **Source:** (https://cloud.google.com/armor/docs/best-practices)
[10d] pattern analyzing data collected by respective distributed agents to identify similar patterns of suspicious activities indicative of an attack or threat to different portions of the computer network,	Google's Backstory service can identify suspicious behavioral patterns. Google Cloud Armor can use Adaptive Protection to analyze network traffic patterns for suspicious activity and create models to detect their signatures. Google Chronicle's VirusTotal service can process and extract patterns from data. The following exemplifies this limitation's existence in Accused Systems:

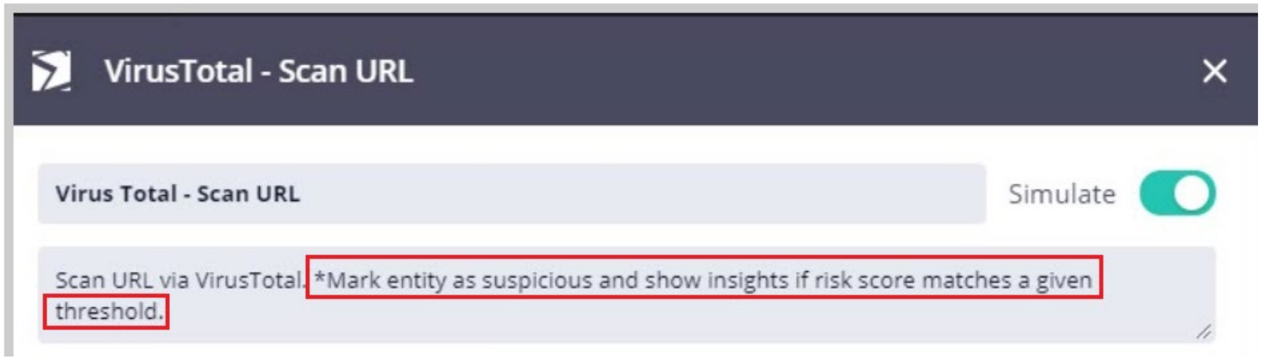
	 In this example, we drill down into the Asset View for the endpoint that accessed the APT domain and can quickly see a beaconing pattern to rare (low prevalence) domains graphically as well as a suspicious process in the timeline panel (ghost.exe) that is launched immediately after. Subsequently, a file (notepab.exe) is written to disk and a new registry run key is created to gain persistence across reboots. Backstory uses stateful URLs so at any time an analyst can capture the exact set of filtering conditions and pass it on to the next analyst for continued triage across shifts or escalation tiers. Similarly, the timeline of events can be easily filtered on data model dimensions and attributes and the relevant set of events can be easily exported.  Analysts can also pivot to the Backstory User View which provides context on who the user is (from directory sources like Active Directory) and whether their behavior is anomalous. User View operates on distinct data sources and events such as Windows (login events), Office 365, and Azure AD. In this example, we see that Todd Fields is a VP of Finance and that there are a few user centric security alerts associated with his account. The graph tells us that while Todd normally only logs into his account between 7am and 5pm, after the compromise of his laptop there has been anomalous account activity past 11pm. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)  Adaptive Protection helps you protect your applications and services from L7 distributed denial-of-service (DDoS) attacks by analyzing patterns of traffic to your backend services, detecting and alerting on suspected attacks, and generating suggested WAF rules to mitigate such attacks. These rules can be tuned to meet your needs. Adaptive Protection can be enabled on a per- security policy basis, but it requires an active Managed Protection subscription in the project. **Source:** (https://cloud.google.com/armor/docs/cloud-armor-overview)  • Adaptive Protection builds several models to detect potential attacks and identify their signatures. The signals used by these models to determine if an attack is ongoing are derived from the observed metadata of incoming request traffic from your projects. Such metadata includes: source IP address, source geography, and the values of some HTTP request headers. **Source:** (https://cloud.google.com/armor/docs/adaptive-protection-overview)

	 It is an easy next step to create a script that uses VirusTotal's Private API to process this data and extract patterns and commonalities. These allow us to add more detail to the adversary profile under construction. The script can launch the same file-similarity search above, making use of the search API. For each match, it can retrieve all sample details programmatically with the file lookup API. In this way, we can automatically discover that the attackers are using other spam campaigns with similar malicious payloads and different subject lines and bodies: **Source:** (https://go.chronicle.security/hubfs/VirusTotal_Enterprise_WP.pdf)
[10e] determining from said pattern analysis whether a probability threshold of suspicious activity indicative of an attack or threat to said computer network has been exceeded,	Google's Backstory service can provide insight on the probability of a compromised device. Google Cloud Armor can provide a confidence level for the likelihood that a change is anomalous. Siemplify SOAR can use VirusTotal to scan URLs and mark them as suspicious if their risk score matches a certain threshold. The following exemplifies this limitation's existence in Accused Systems:  The combination of his role, recent account related alerts from 3rd party SaaS applications (which may also come from a CASB such as Netskope), and anomalous access by time of day suggest Todd's account may have been compromised. The likelihood is only higher because we know Todd's endpoint has been compromised by an APT that successfully downloaded a malicious payload to his laptop **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)

You can tune Adaptive Protection alerts based on several metrics. The alerts, which are sent to Cloud Logging, include a *confidence level*, an *attack signature*, a suggested rule, and an estimated *impacted baseline rate* associated with the suggested rule.

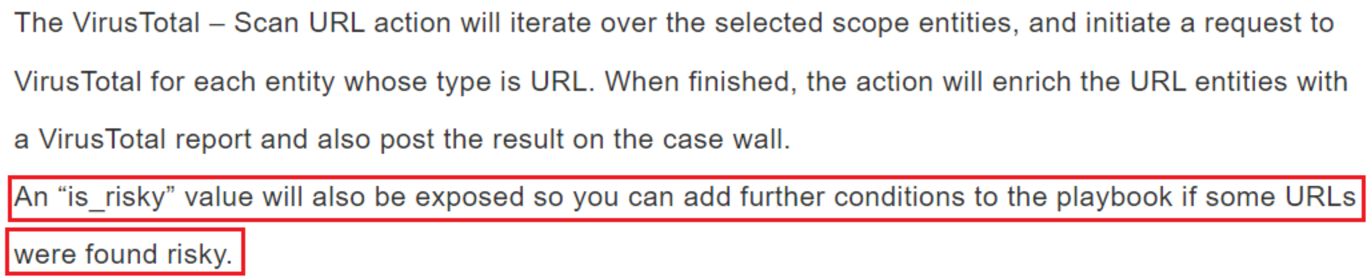
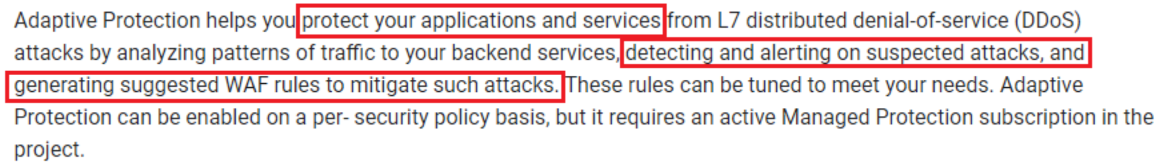
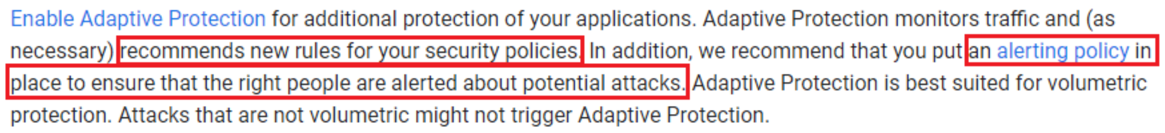
- The confidence level indicates the confidence with which Adaptive Protection models predict that the observed change in traffic pattern is anomalous.
- The impacted baseline rates associated with the suggested rule represent the percentage of existing baseline traffic that is caught by the rule. Two rates are provided. The first is the percentage relative to the traffic into the specific backend service under attack. The second is the percentage relative to all the traffic going through the security policy including all configured backend service targets (not just the one being attacked).

Source: (https://cloud.google.com/armor/docs/adaptive-protection-overview)



Scan URL via VirusTotal. *Mark entity as suspicious and show insights if risk score matches a given threshold.

Source: (https://documents.siemplify.co/en/articles/417-working-with-playbook-simulator)

	 The VirusTotal – Scan URL action will iterate over the selected scope entities, and initiate a request to VirusTotal for each entity whose type is URL. When finished, the action will enrich the URL entities with a VirusTotal report and also post the result on the case wall. An "is_risky" value will also be exposed so you can add further conditions to the playbook if some URLs were found risky. **Source:** (https://documents.siemplify.co/en/articles/481-how-do-i-scan-multiple-urls-in-virustotal)
[10f] and when said probability threshold has been exceeded by said similar patterns of suspicious activities, alerting other agents, a central server, and/or a human operator.	Google Cloud Armor can use Adaptive Protection to detect and alert on suspected attacks. It can generate events with a confidence score indicating the likelihood of the pattern change as anomalous and provide potential mitigation methods. The following exemplifies this limitation's existence in Accused Systems:  Adaptive Protection helps you protect your applications and services from L7 distributed denial-of-service (DDoS) attacks by analyzing patterns of traffic to your backend services, detecting and alerting on suspected attacks, and generating suggested WAF rules to mitigate such attacks. These rules can be tuned to meet your needs. Adaptive Protection can be enabled on a per- security policy basis, but it requires an active Managed Protection subscription in the project. **Source:** (https://cloud.google.com/armor/docs/cloud-armor-overview)  Enable Adaptive Protection for additional protection of your applications. Adaptive Protection monitors traffic and (as necessary) recommends new rules for your security policies. In addition, we recommend that you put an alerting policy in place to ensure that the right people are alerted about potential attacks. Adaptive Protection is best suited for volumetric protection. Attacks that are not volumetric might not trigger Adaptive Protection. **Source:** (https://cloud.google.com/armor/docs/best-practices)

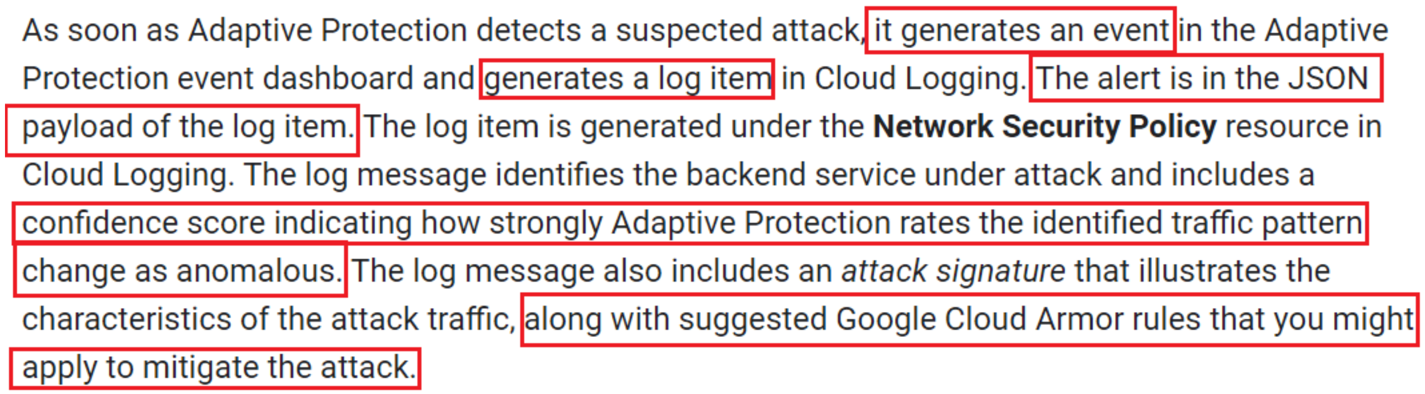
	 As soon as Adaptive Protection detects a suspected attack, it generates an event in the Adaptive Protection event dashboard and generates a log item in Cloud Logging. The alert is in the JSON payload of the log item. The log item is generated under the **Network Security Policy** resource in Cloud Logging. The log message identifies the backend service under attack and includes a confidence score indicating how strongly Adaptive Protection rates the identified traffic pattern change as anomalous. The log message also includes an *attack signature* that illustrates the characteristics of the attack traffic, along with suggested Google Cloud Armor rules that you might apply to mitigate the attack. **Source:** (https://cloud.google.com/armor/docs/adaptive-protection-overview)

Exhibit G

to

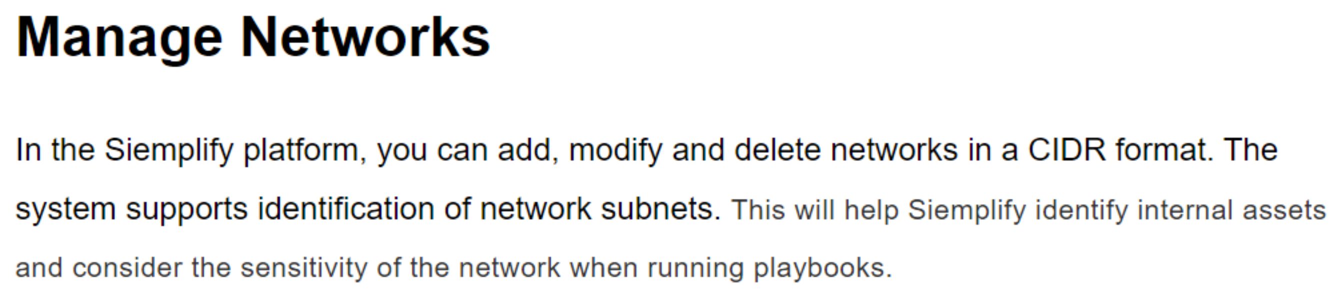
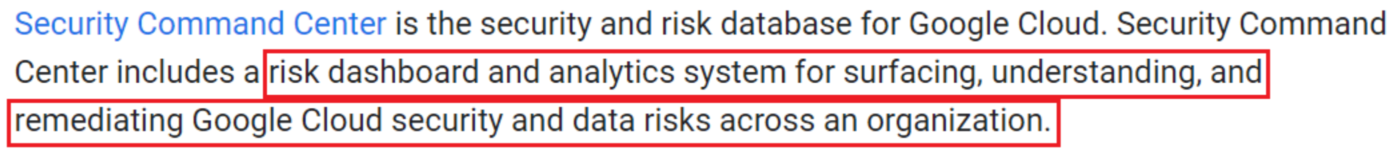
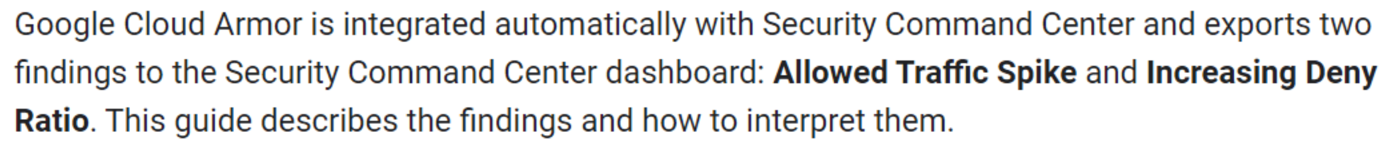
Complaint for Patent Infringement

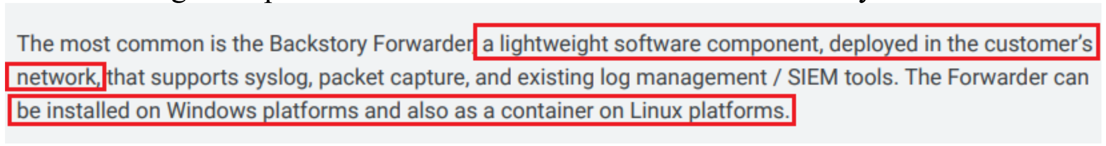
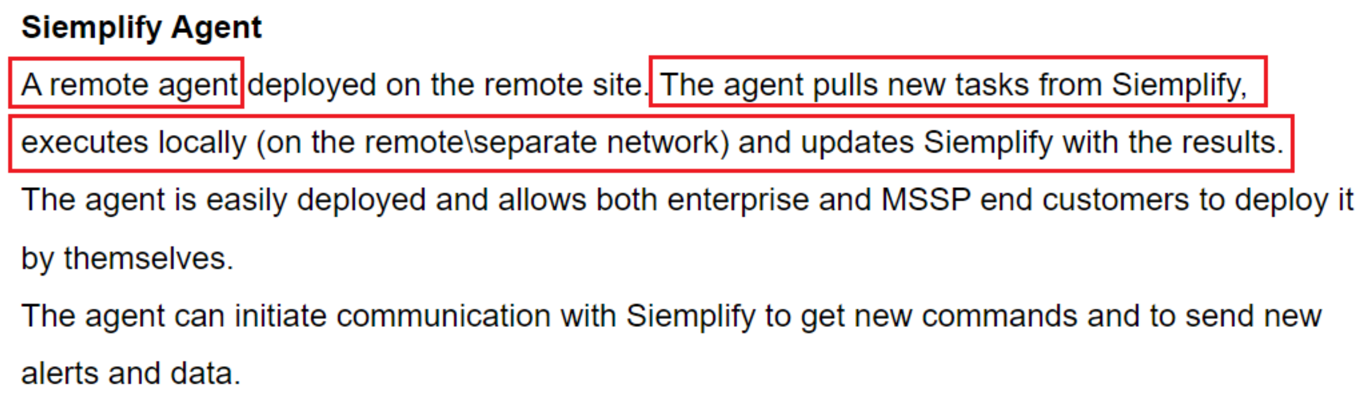
Claim Chart[1] for the ’470 Patent

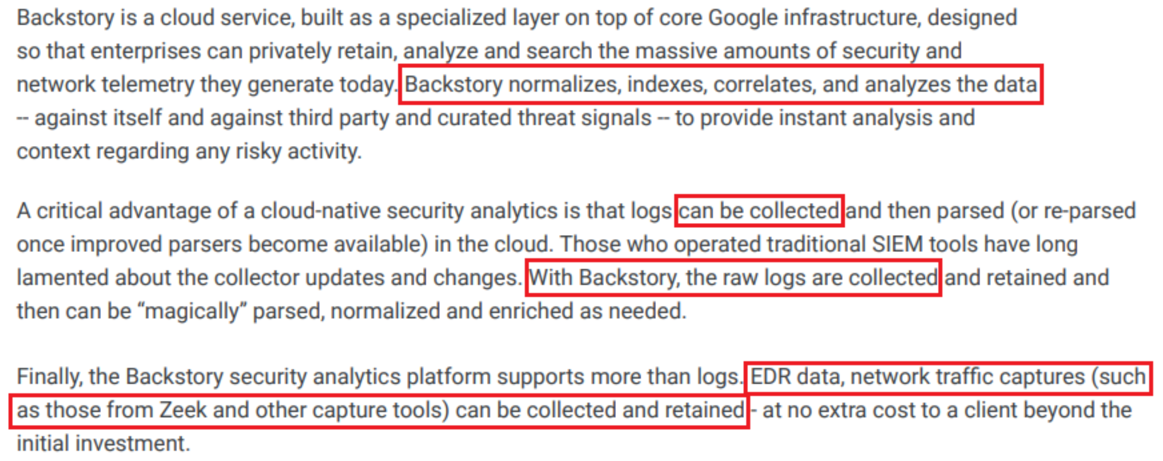
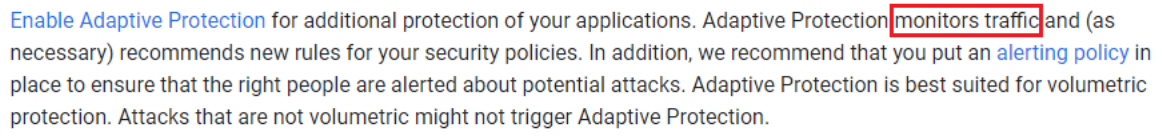
[1] Plaintiff provides this exemplary claim chart for the purposes of showing one basis of infringement of one of the Patents-in-suit by Defendant’s Accused Products as defined in the Complaint. This exemplary claim chart addresses the Accused Products broadly based on the fact that the Accused Products infringe in the same general way. Plaintiff reserves its right to amend and fully provide its infringement arguments and evidence thereof until its Preliminary and Final Infringement Contentions are later produced according to the court’s scheduling order in this case.

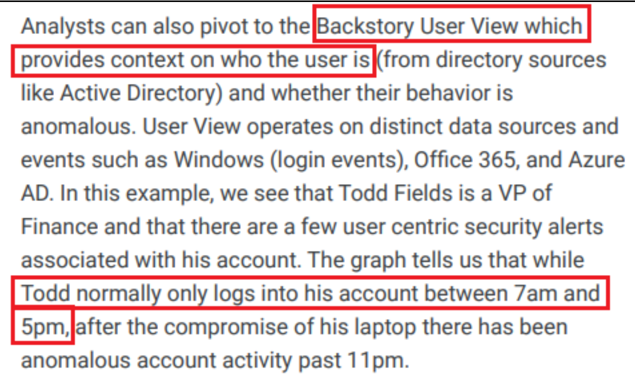
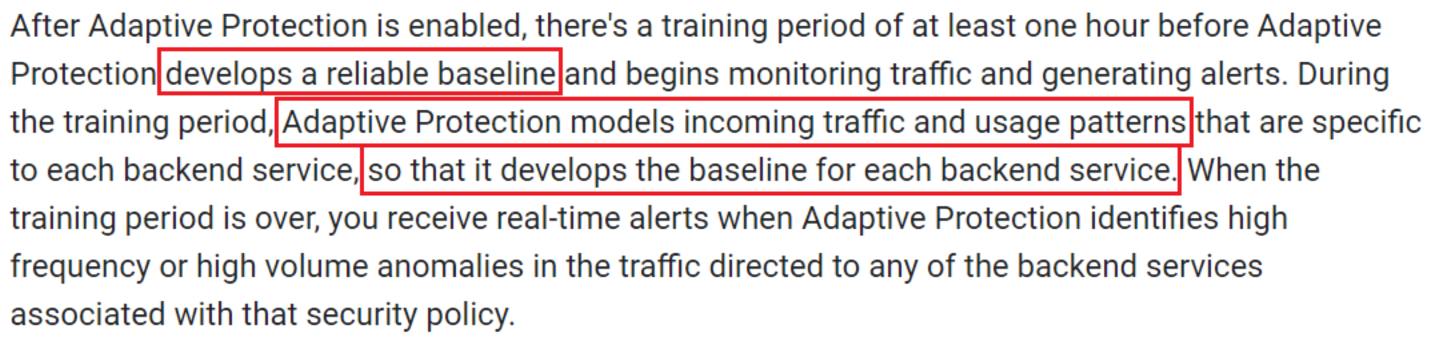
CLAIM CHART

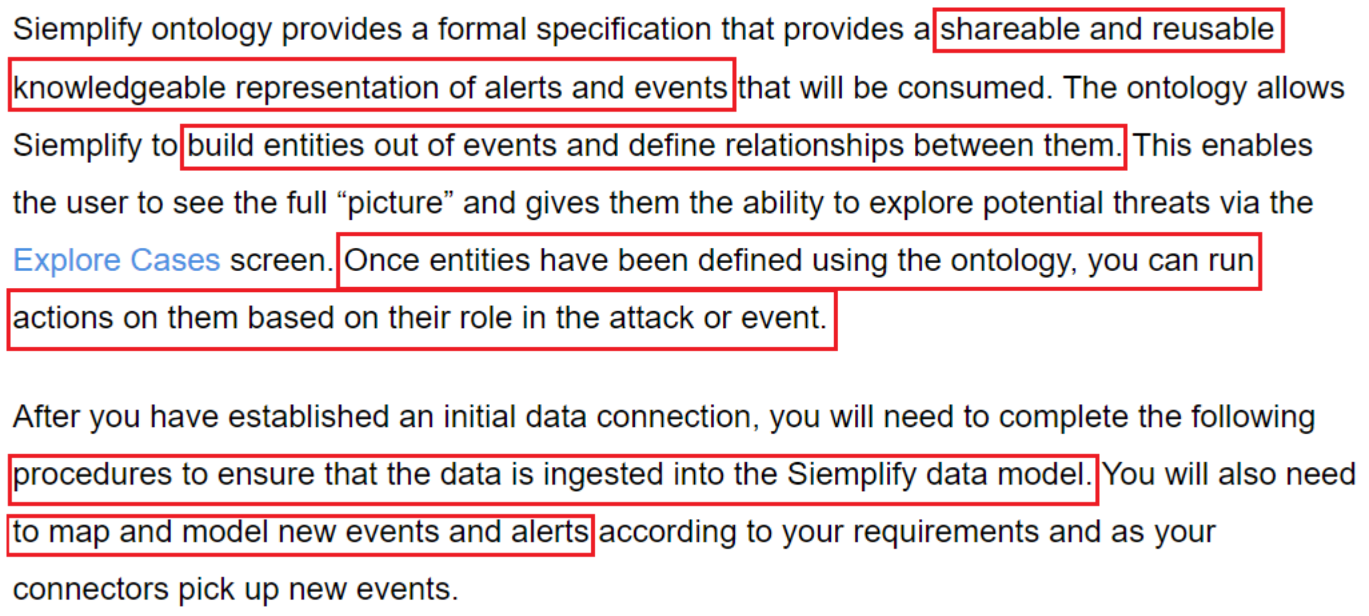
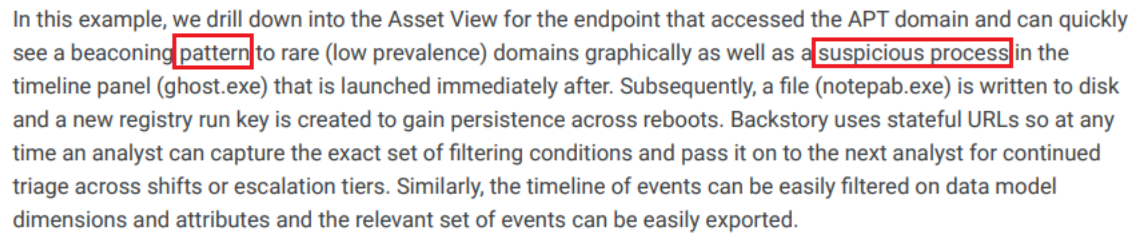
U.S. PATENT NO. 9,503,470 B2 – CLAIM 1

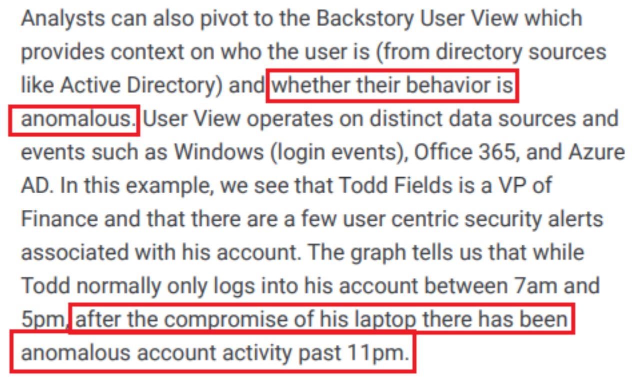
Claim 1	Corresponding Structure in Accused Systems – Google LLC
[1a] A system that detects the state of a computer network, comprising:	Google LLC ("Google") provides a cloud service known as Google Cloud that contains cybersecurity services including Chronicle, Siemplify SOAR, Google Web Risk, and Google Cloud Armor. *See* https://cloud.google.com/products, https://chronicle.security/product/, https://cloud.google.com/solutions/security-orchestration-automation-response, https://cloud.google.com/web-risk, and https://cloud.google.com/armor. Siemplify SOAR can allow users to manage networks and identify subnets. Google Cloud Armor can integrate with Google's Security Command Center to view security and data risks across the organization.  **Manage Networks** In the Siemplify platform, you can add, modify and delete networks in a CIDR format. The system supports identification of network subnets. This will help Siemplify identify internal assets and consider the sensitivity of the network when running playbooks. **Source:** (https://documents.siemplify.co/en/articles/451-manage-networks)  Security Command Center is the security and risk database for Google Cloud. Security Command Center includes a risk dashboard and analytics system for surfacing, understanding, and remediating Google Cloud security and data risks across an organization.  Google Cloud Armor is integrated automatically with Security Command Center and exports two findings to the Security Command Center dashboard: **Allowed Traffic Spike** and **Increasing Deny Ratio**. This guide describes the findings and how to interpret them. **Source:** (https://cloud.google.com/armor/docs/cscc-findings)

	Google Cloud together with various equipment, services, components, and/or software utilized in providing Google Cloud collectively include a system and method for a distributed agent based model for security monitoring and response as described by the meaning of this claim. Google Cloud is made available by a system owned and/or operated by Google. “Because infringement liability is not dependent on ownership, e.g., use of a system can infringe (35 U.S.C. § 271), infringement is not dependent on ownership of all limitations of a claim.”
[1b] a plurality of distributed agents disposed in said computer network, each said distributed agent including a microprocessor adapted to:	Google’s Backstory service can use the Backstory Forwarder as an agent to collect network data. Siemplify SOAR can use remote agents deployed on customer networks. The following exemplifies this limitation’s existence in Accused Systems:  The most common is the Backstory Forwarder a lightweight software component, deployed in the customer’s network, that supports syslog, packet capture, and existing log management / SIEM tools. The Forwarder can be installed on Windows platforms and also as a container on Linux platforms. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)  **Siemplify Agent** A remote agent deployed on the remote site. The agent pulls new tasks from Siemplify, executes locally (on the remote\separate network) and updates Siemplify with the results. The agent is easily deployed and allows both enterprise and MSSP end customers to deploy it by themselves. The agent can initiate communication with Siemplify to get new commands and to send new alerts and data. **Source:** (https://documents.siemplify.co/en/articles/4-remote-agents-for-cloud-overview)

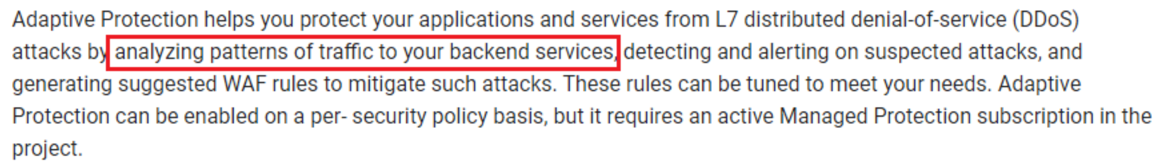
[1c] passively collect, monitor, and aggregate data representative of activities of respective nodes within said computer network,	Google's Backstory service can collect, correlate, and analyze telemetry data and logs. Google Cloud Armor can use Adaptive Protection to monitor traffic. The following exemplifies this limitation's existence in Accused Systems:  Backstory is a cloud service, built as a specialized layer on top of core Google infrastructure, designed so that enterprises can privately retain, analyze and search the massive amounts of security and network telemetry they generate today. Backstory normalizes, indexes, correlates, and analyzes the data -- against itself and against third party and curated threat signals -- to provide instant analysis and context regarding any risky activity. A critical advantage of a cloud-native security analytics is that logs can be collected and then parsed (or re-parsed once improved parsers become available) in the cloud. Those who operated traditional SIEM tools have long lamented about the collector updates and changes. With Backstory, the raw logs are collected and retained and then can be "magically" parsed, normalized and enriched as needed. Finally, the Backstory security analytics platform supports more than logs. EDR data, network traffic captures (such as those from Zeek and other capture tools) can be collected and retained - at no extra cost to a client beyond the initial investment. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)  Enable Adaptive Protection for additional protection of your applications. Adaptive Protection monitors traffic and (as necessary) recommends new rules for your security policies. In addition, we recommend that you put an alerting policy in place to ensure that the right people are alerted about potential attacks. Adaptive Protection is best suited for volumetric protection. Attacks that are not volumetric might not trigger Adaptive Protection. **Source:** (https://cloud.google.com/armor/docs/best-practices)
[1d] analyze collected data to develop activity models representative of activities of said computer network in a normal state and	Google's Backstory service can determine normal usage behavior and provide context through Backstory User View. Google Cloud Armor can use Adaptive Protection which establishes a baseline model of network traffic and usage patterns. Siemplify SOAR can model new events and alerts into shareable and reusable entities through ontology. The following exemplifies this limitation's existence in Accused Systems:

	 Analysts can also pivot to the Backstory User View which provides context on who the user is (from directory sources like Active Directory) and whether their behavior is anomalous. User View operates on distinct data sources and events such as Windows (login events), Office 365, and Azure AD. In this example, we see that Todd Fields is a VP of Finance and that there are a few user centric security alerts associated with his account. The graph tells us that while Todd normally only logs into his account between 7am and 5pm, after the compromise of his laptop there has been anomalous account activity past 11pm. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)  After Adaptive Protection is enabled, there's a training period of at least one hour before Adaptive Protection develops a reliable baseline and begins monitoring traffic and generating alerts. During the training period, Adaptive Protection models incoming traffic and usage patterns that are specific to each backend service, so that it develops the baseline for each backend service. When the training period is over, you receive real-time alerts when Adaptive Protection identifies high frequency or high volume anomalies in the traffic directed to any of the backend services associated with that security policy. **Source:** (https://cloud.google.com/armor/docs/adaptive-protection-overview)

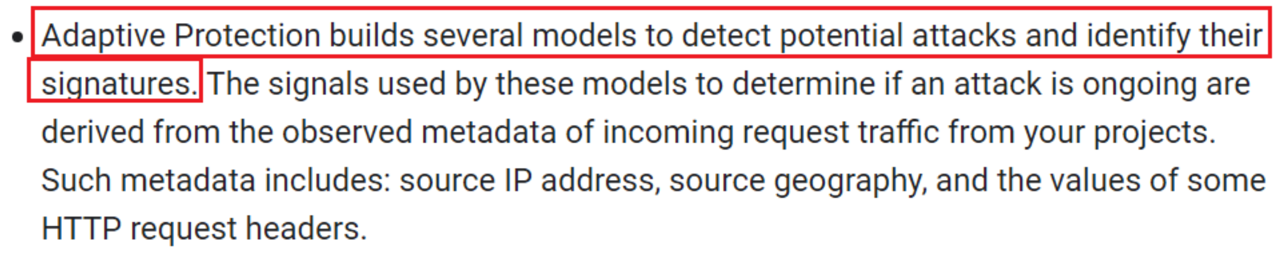
	 Siemplify ontology provides a formal specification that provides a shareable and reusable knowledgeable representation of alerts and events that will be consumed. The ontology allows Siemplify to build entities out of events and define relationships between them. This enables the user to see the full “picture” and gives them the ability to explore potential threats via the Explore Cases screen. Once entities have been defined using the ontology, you can run actions on them based on their role in the attack or event. After you have established an initial data connection, you will need to complete the following procedures to ensure that the data is ingested into the Siemplify data model. You will also need to map and model new events and alerts according to your requirements and as your connectors pick up new events. **Source:** (https://documents.siemplify.co/en/articles/454-ontology-overview)
[1e] activities of said computer network in an abnormal state as a result of intrusions, infections, scams, code emulating code or humans, and/or other suspicious activities in said computer network, and	Google’s Backstory service can identify suspicious behavioral patterns. Google Cloud Armor can use Adaptive Protection to analyze network traffic patterns for suspicious activity and create models to detect their signatures. Google Chronicle’s VirusTotal service can process and extract patterns from data. The following exemplifies this limitation’s existence in Accused Systems:  In this example, we drill down into the Asset View for the endpoint that accessed the APT domain and can quickly see a beaconing pattern to rare (low prevalence) domains graphically as well as a suspicious process in the timeline panel (ghost.exe) that is launched immediately after. Subsequently, a file (notepab.exe) is written to disk and a new registry run key is created to gain persistence across reboots. Backstory uses stateful URLs so at any time an analyst can capture the exact set of filtering conditions and pass it on to the next analyst for continued triage across shifts or escalation tiers. Similarly, the timeline of events can be easily filtered on data model dimensions and attributes and the relevant set of events can be easily exported.


Analysts can also pivot to the Backstory User View which provides context on who the user is (from directory sources like Active Directory) and whether their behavior is anomalous. User View operates on distinct data sources and events such as Windows (login events), Office 365, and Azure AD. In this example, we see that Todd Fields is a VP of Finance and that there are a few user centric security alerts associated with his account. The graph tells us that while Todd normally only logs into his account between 7am and 5pm, after the compromise of his laptop there has been anomalous account activity past 11pm.

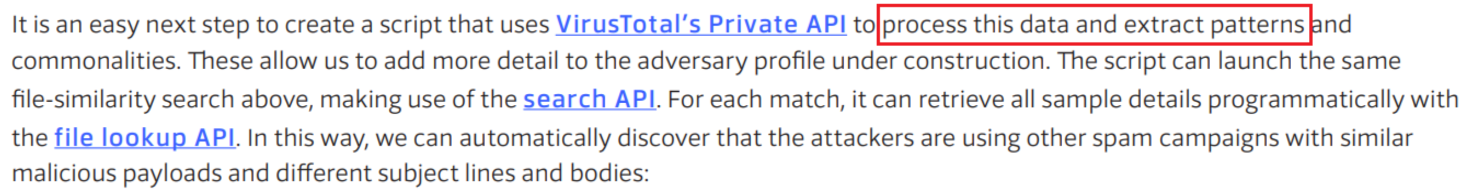
Source: (https://go.chronicle.security/hubfs/Backstory_WP.pdf)


Adaptive Protection helps you protect your applications and services from L7 distributed denial-of-service (DDoS) attacks by analyzing patterns of traffic to your backend services, detecting and alerting on suspected attacks, and generating suggested WAF rules to mitigate such attacks. These rules can be tuned to meet your needs. Adaptive Protection can be enabled on a per- security policy basis, but it requires an active Managed Protection subscription in the project.

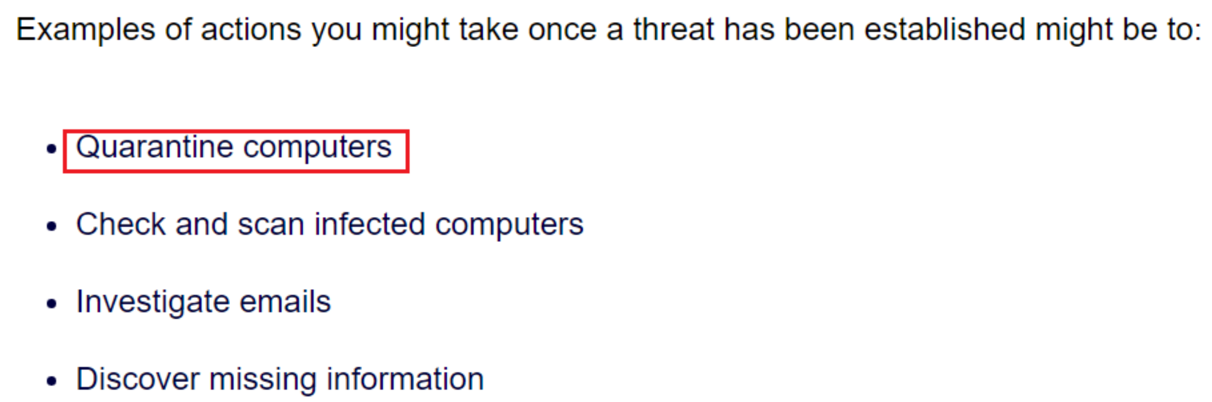
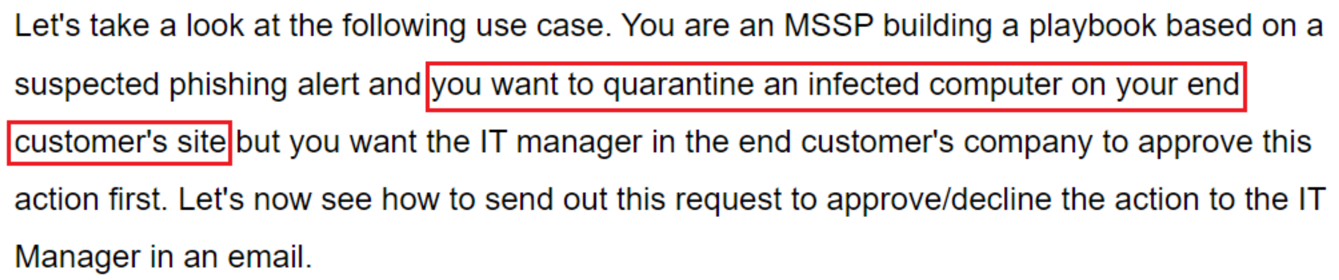
Source: (https://cloud.google.com/armor/docs/cloud-armor-overview)

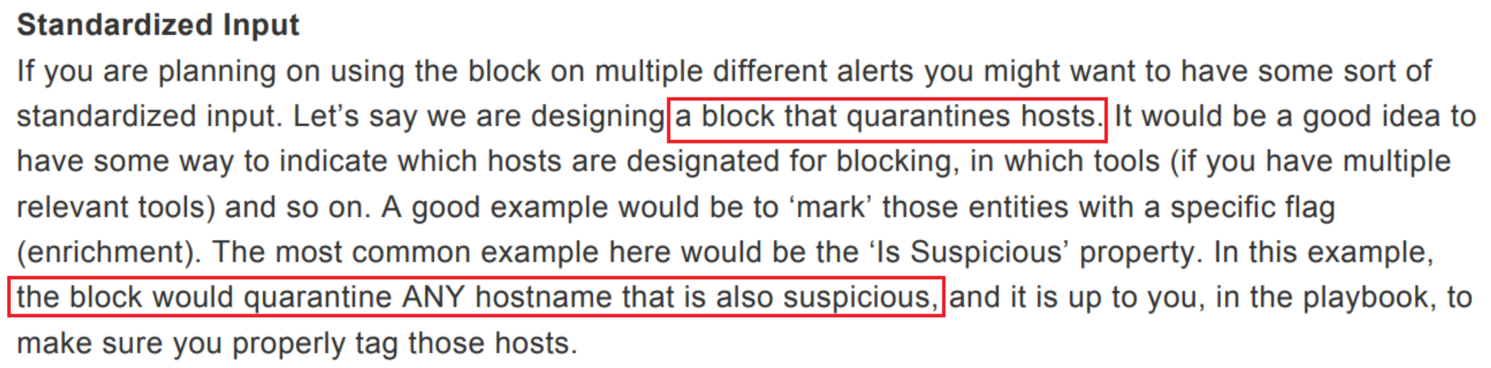

- Adaptive Protection builds several models to detect potential attacks and identify their signatures. The signals used by these models to determine if an attack is ongoing are derived from the observed metadata of incoming request traffic from your projects. Such metadata includes: source IP address, source geography, and the values of some HTTP request headers.

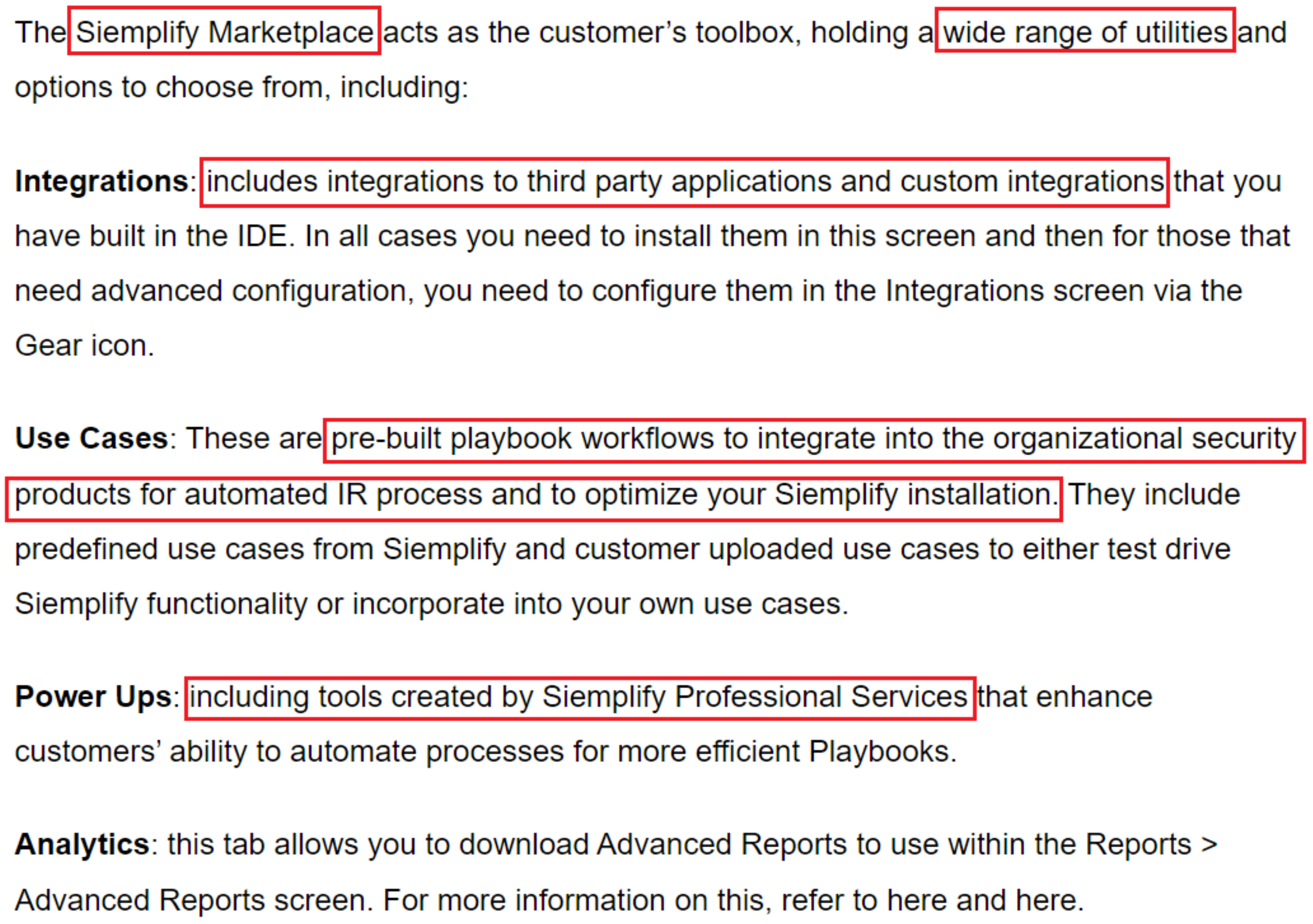
Source: (https://cloud.google.com/armor/docs/adaptive-protection-overview)

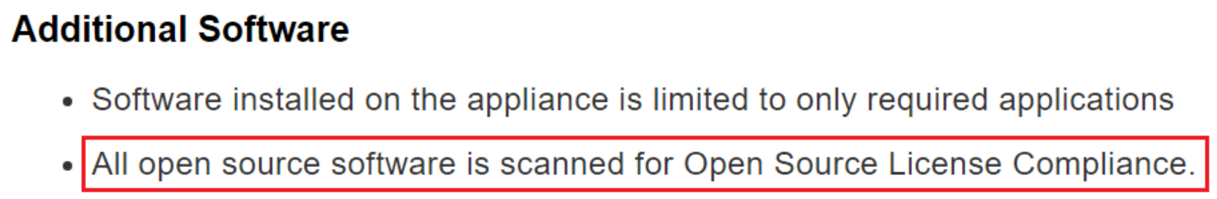
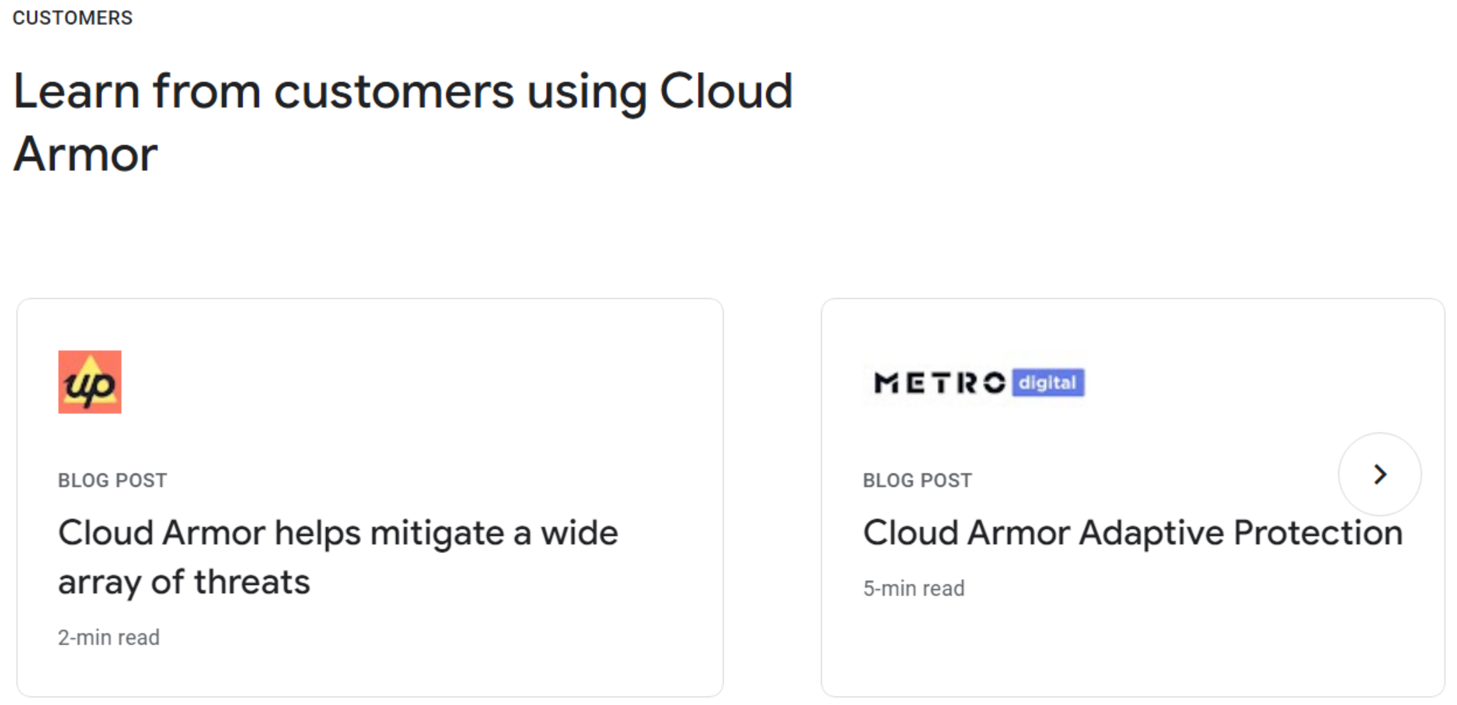

It is an easy next step to create a script that uses VirusTotal's Private API to process this data and extract patterns and commonalities. These allow us to add more detail to the adversary profile under construction. The script can launch the same file-similarity search above, making use of the search API. For each match, it can retrieve all sample details programmatically with the file lookup API. In this way, we can automatically discover that the attackers are using other spam campaigns with similar malicious payloads and different subject lines and bodies:

Source: (https://go.chronicle.security/hubfs/VirusTotal_Enterprise_WP.pdf)

[1f] generate counter-offensive measures where unauthorized access to a program or file containing executable code results in the program or file disabling an operating system with all associated applications of a computer in the computer network until/unless the presumed attacker is able to prove to the machine owner/victim that the presumed attacker had been authorized to access the target data or machine provoking the said counter offensive measure; and	Siemplify SOAR can quarantine infected endpoint computers. The following exemplifies this limitation's existence in Accused Systems:  Examples of actions you might take once a threat has been established might be to: • Quarantine computers • Check and scan infected computers • Investigate emails • Discover missing information **Source:** (https://documents.siemplify.co/en/articles/443-explore-entities-and-alerts-investigation)  Let's take a look at the following use case. You are an MSSP building a playbook based on a suspected phishing alert and you want to quarantine an infected computer on your end customer's site but you want the IT manager in the end customer's company to approve this action first. Let's now see how to send out this request to approve/decline the action to the IT Manager in an email. **Source:** (https://documents.siemplify.co/en/articles/718-assign-approval-links-in-actions)

	 **Standardized Input** If you are planning on using the block on multiple different alerts you might want to have some sort of standardized input. Let's say we are designing a block that quarantines hosts. It would be a good idea to have some way to indicate which hosts are designated for blocking, in which tools (if you have multiple relevant tools) and so on. A good example would be to 'mark' those entities with a specific flag (enrichment). The most common example here would be the 'Is Suspicious' property. In this example, the block would quarantine ANY hostname that is also suspicious, and it is up to you, in the playbook, to make sure you properly tag those hosts. **Source:** (https://cdn.elev.io/file/uploads/raEXHEUeZuVTHs6dRYCY6y2voCpa34wo3QprCO0wQX4/81jDpl5Mb06o-0gjVrwAwU0Yw-jNp2YRDEFHDVUHd6U/PlaybookTheoryof--9Y.pdf)
[1g] a server that provides a security and validity score for free software available for download, the validity score comprising three components including	Siemplify SOAR contains the Siemplify Marketplace which can allow users to choose from a wide range of utilities. These include integrations to third party applications, pre-made use cases, and other tools. The following exemplifies this limitation's existence in Accused Systems:

	 The Siemplify Marketplace acts as the customer's toolbox, holding a wide range of utilities and options to choose from, including: **Integrations**: includes integrations to third party applications and custom integrations that you have built in the IDE. In all cases you need to install them in this screen and then for those that need advanced configuration, you need to configure them in the Integrations screen via the Gear icon. **Use Cases**: These are pre-built playbook workflows to integrate into the organizational security products for automated IR process and to optimize your Siemplify installation. They include predefined use cases from Siemplify and customer uploaded use cases to either test drive Siemplify functionality or incorporate into your own use cases. **Power Ups**: including tools created by Siemplify Professional Services that enhance customers' ability to automate processes for more efficient Playbooks. **Analytics**: this tab allows you to download Advanced Reports to use within the Reports > Advanced Reports screen. For more information on this, refer to here and here. **Source:** (https://documents.siemplify.co/en/articles/73-using-the-marketplace)

[1h] a first component computed based on security of the free software itself,	Siemplify SOAR can scan open source software for Open Source License Compliance. The following exemplifies this limitation's existence in Accused Systems:  **Additional Software** • Software installed on the appliance is limited to only required applications • All open source software is scanned for Open Source License Compliance. **Source:** (https://documents.siemplify.co/en/articles/321-hardening-and-security-procedures)
[1i] a second component computed based on experiences users have with the free software, and	Google Cloud contains customer case studies and explanations for why the customer chose the service. The following exemplifies this limitation's existence in Accused Systems:  CUSTOMERS Learn from customers using Cloud Armor BLOG POST Cloud Armor helps mitigate a wide array of threats 2-min read BLOG POST Cloud Armor Adaptive Protection 5-min read

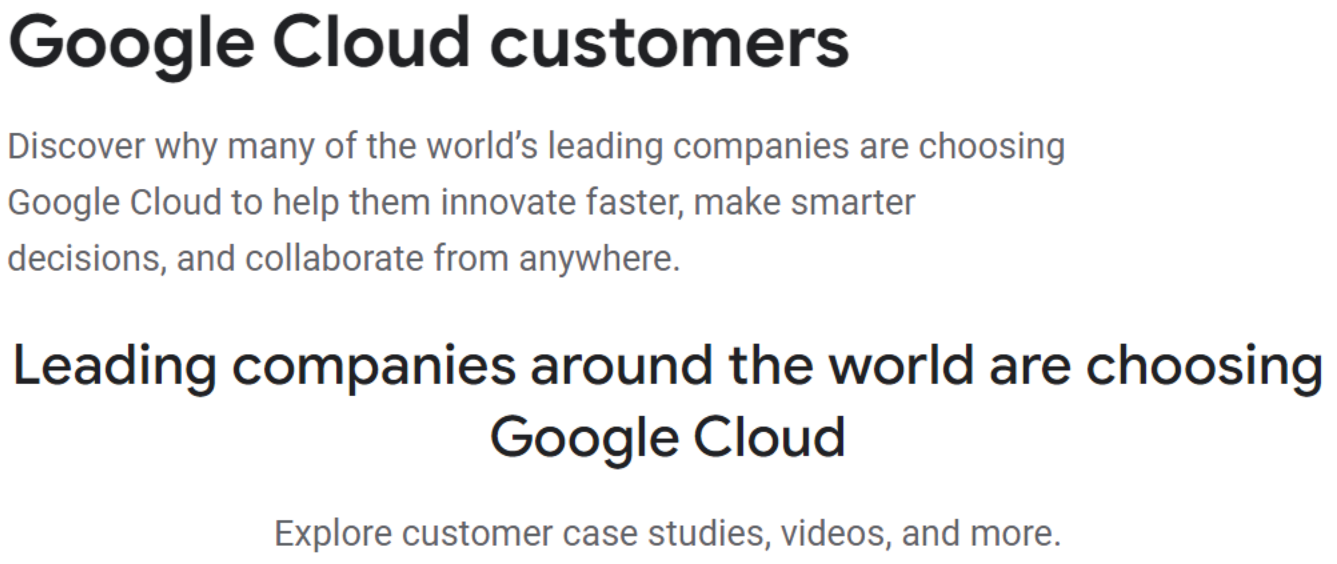
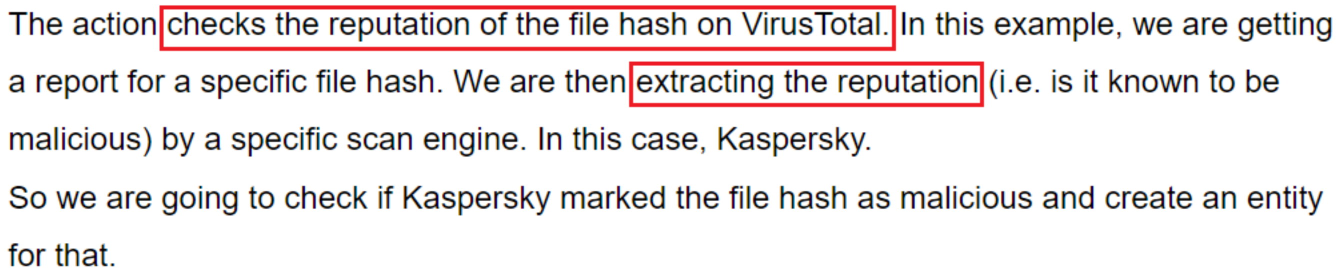
	Source: (https://cloud.google.com/armor)  Google Cloud customers Discover why many of the world's leading companies are choosing Google Cloud to help them innovate faster, make smarter decisions, and collaborate from anywhere. Leading companies around the world are choosing Google Cloud Explore customer case studies, videos, and more. **Source:** (https://cloud.google.com/customers)
[1j] a third component based on a reputation of a programmer that created the free software.	Siemplify SOAR can check the reputation of a file hash using VirusTotal. The following exemplifies this limitation's existence in Accused Systems:  The action checks the reputation of the file hash on VirusTotal. In this example, we are getting a report for a specific file hash. We are then extracting the reputation (i.e. is it known to be malicious) by a specific scan engine. In this case, Kaspersky. So we are going to check if Kaspersky marked the file hash as malicious and create an entity for that. **Source:** (https://documents.siemplify.co/en/articles/511-use-cases-for-expression-builder)

Exhibit H

to

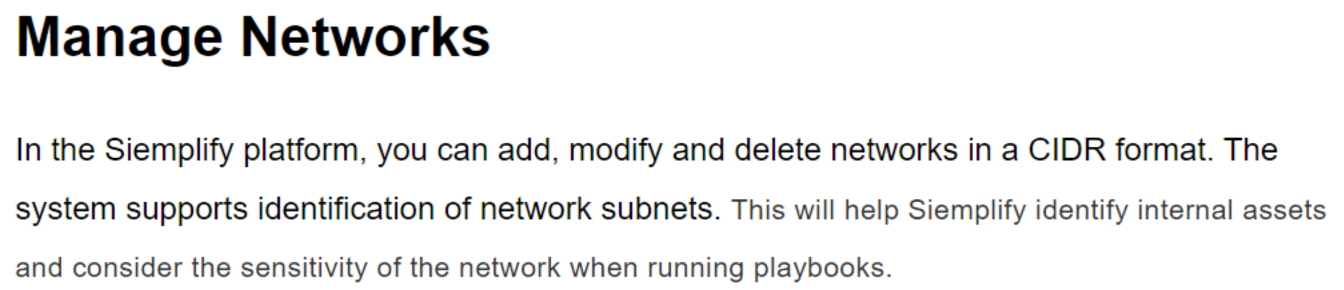
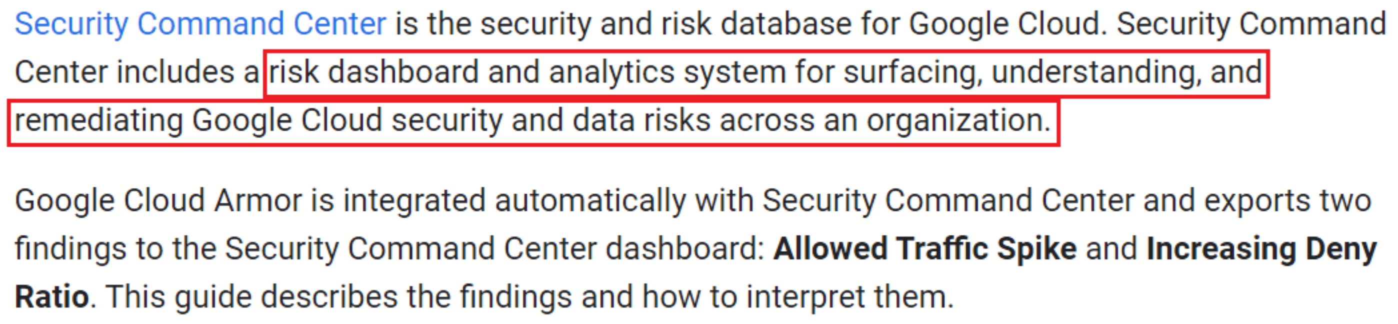
Complaint for Patent Infringement

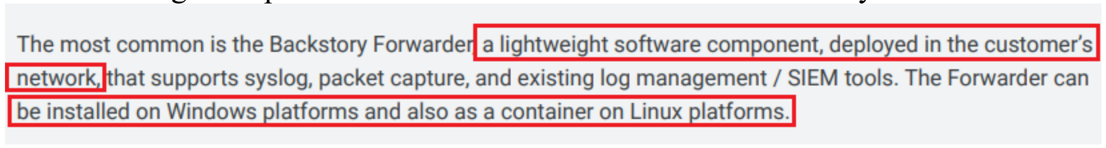
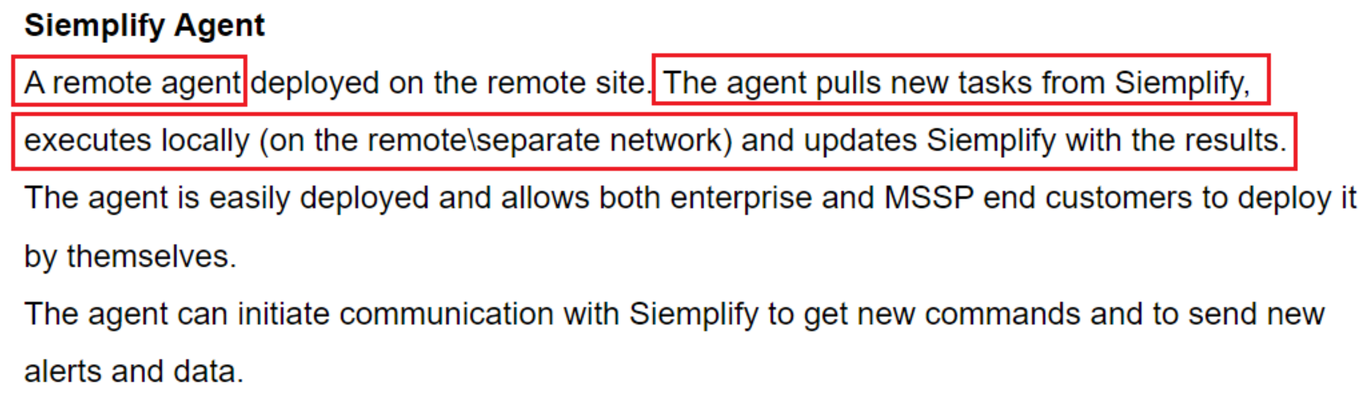
Claim Chart[1] for the ’974 Patent

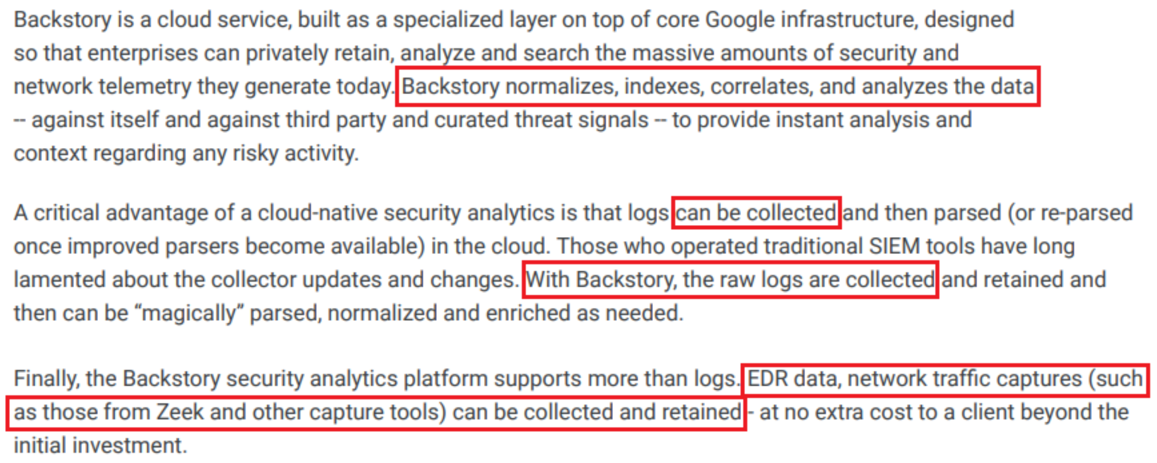
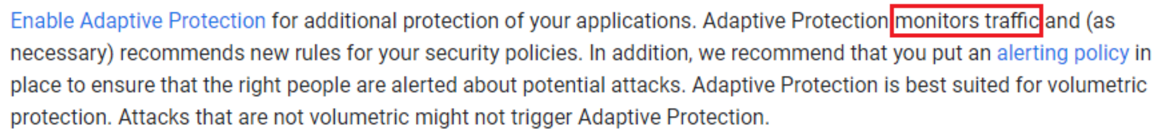
[1] Plaintiff provides this exemplary claim chart for the purposes of showing one basis of infringement of one of the Patents-in-suit by Defendant’s Accused Products as defined in the Complaint. This exemplary claim chart addresses the Accused Products broadly based on the fact that the Accused Products infringe in the same general way. Plaintiff reserves its right to amend and fully provide its infringement arguments and evidence thereof until its Preliminary and Final Infringement Contentions are later produced according to the court’s scheduling order in this case.

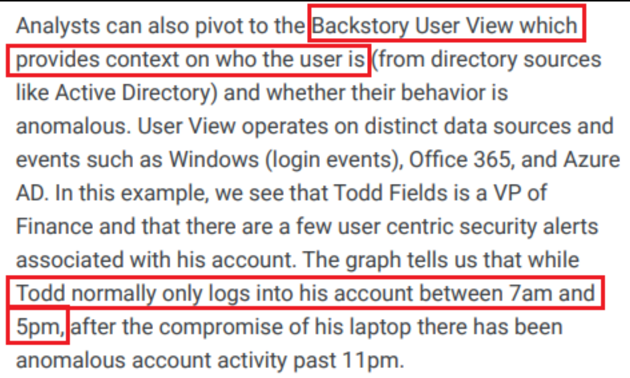
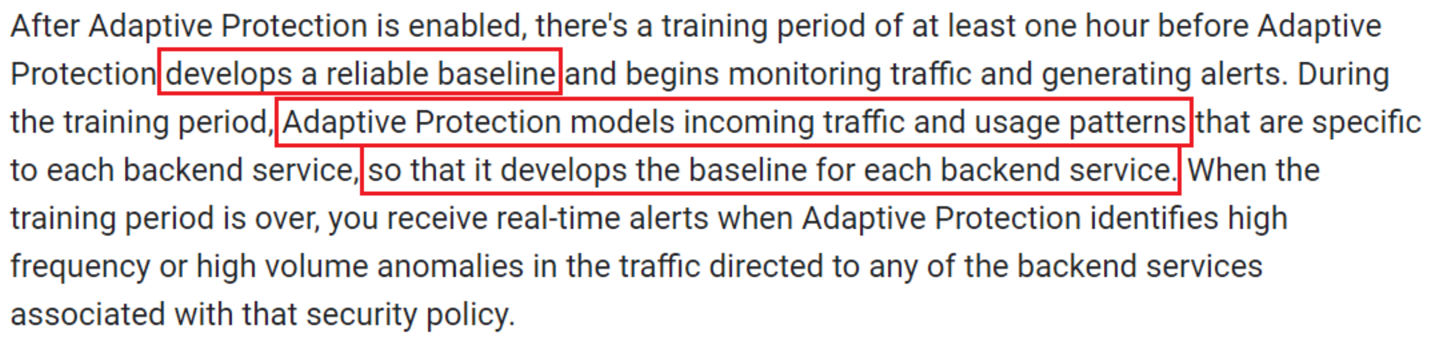
CLAIM CHART

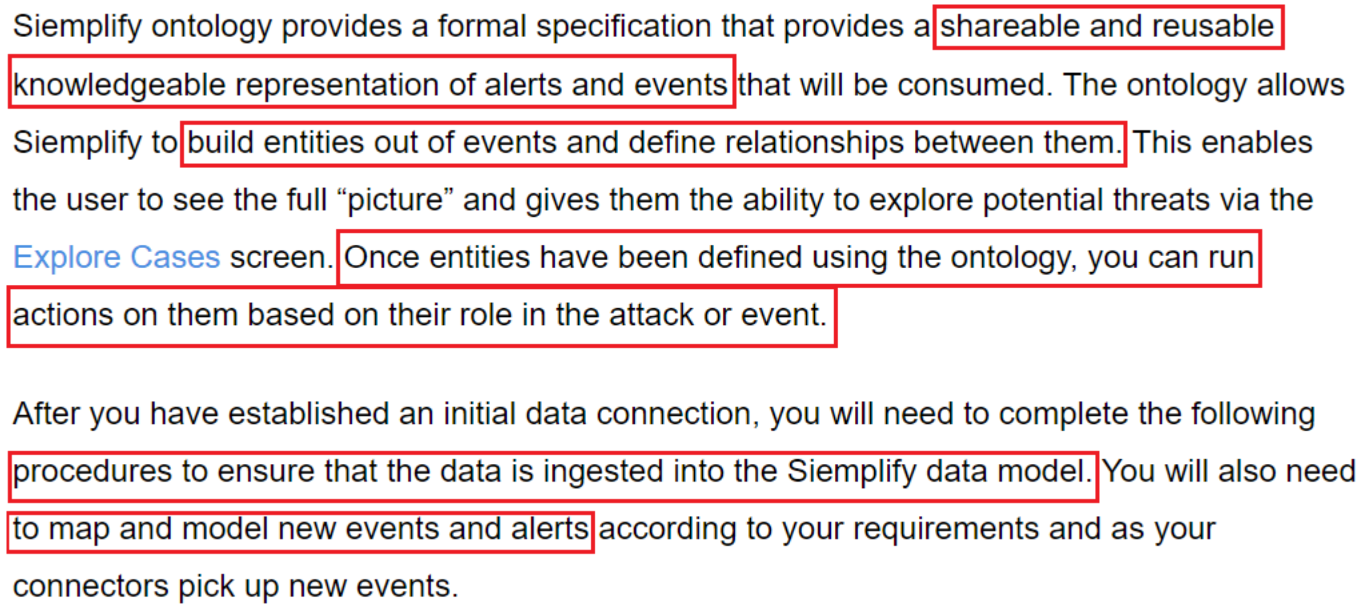
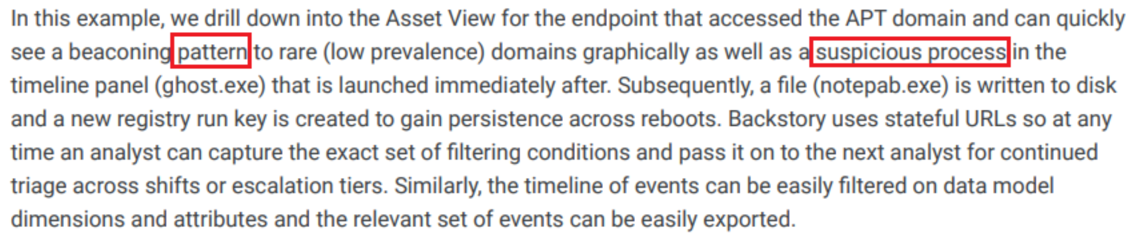
U.S. PATENT NO. 11,171,974 B2 – CLAIM 1

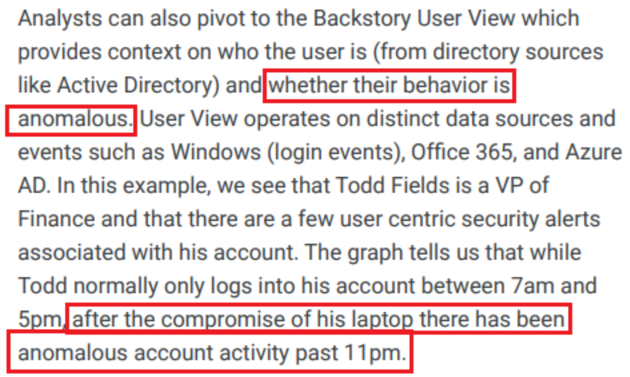
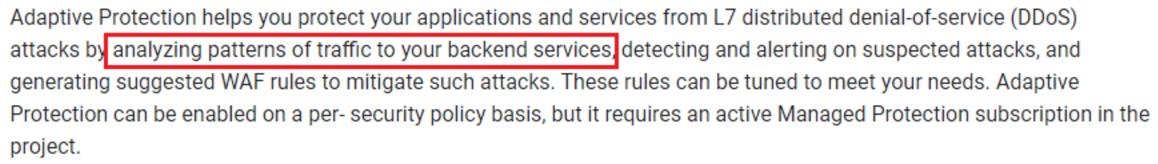
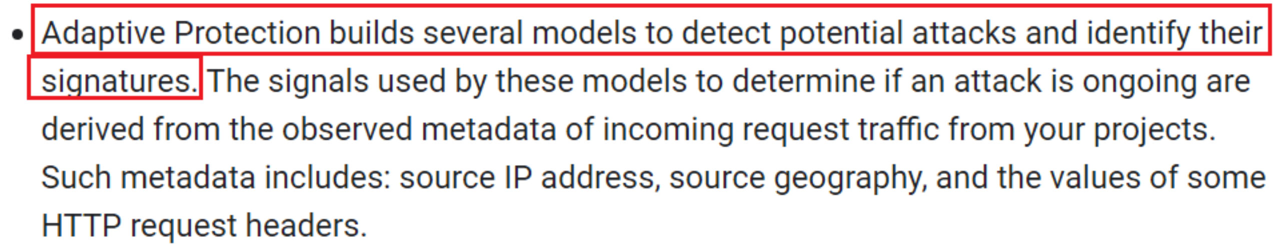
Claim 1	Corresponding Structure in Accused Systems – Google LLC
[1a] A system that detects the state of a computer network, comprising:	Google LLC ("Google") provides a cloud service known as Google Cloud that contains cybersecurity services including Chronicle, Siemplify SOAR, Google Web Risk, and Google Cloud Armor. *See* https://cloud.google.com/products, https://chronicle.security/product/, https://cloud.google.com/solutions/security-orchestration-automation-response, https://cloud.google.com/web-risk, and https://cloud.google.com/armor. Siemplify SOAR can allow users to manage networks and identify subnets. Google Cloud Armor can integrate with Google's Security Command Center to view security and data risks across the organization.  **Manage Networks** In the Siemplify platform, you can add, modify and delete networks in a CIDR format. The system supports identification of network subnets. This will help Siemplify identify internal assets and consider the sensitivity of the network when running playbooks. **Source:** (https://documents.siemplify.co/en/articles/451-manage-networks)  Security Command Center is the security and risk database for Google Cloud. Security Command Center includes a risk dashboard and analytics system for surfacing, understanding, and remediating Google Cloud security and data risks across an organization. Google Cloud Armor is integrated automatically with Security Command Center and exports two findings to the Security Command Center dashboard: **Allowed Traffic Spike** and **Increasing Deny Ratio**. This guide describes the findings and how to interpret them. **Source:** (https://cloud.google.com/armor/docs/cscc-findings)

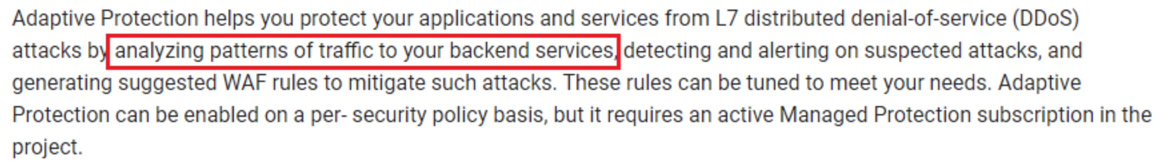
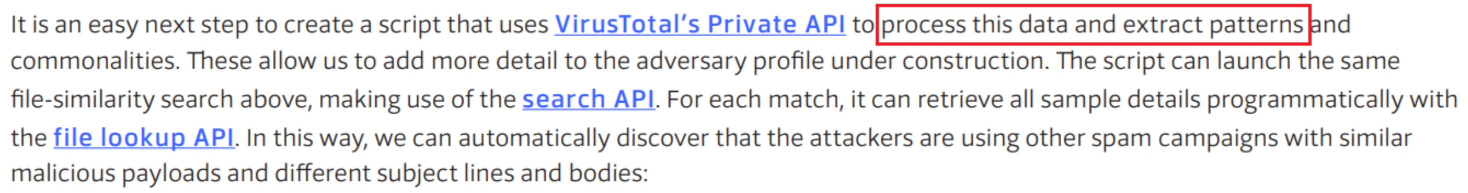
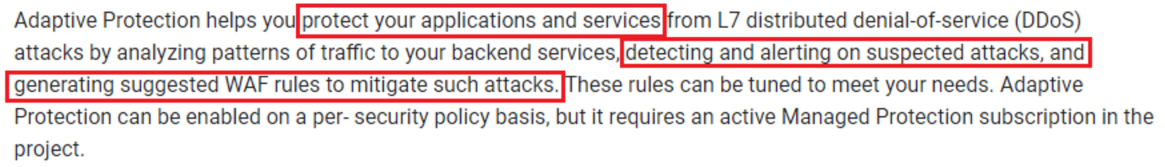
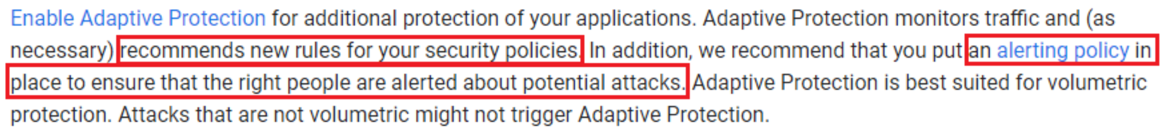
	Google Cloud together with various equipment, services, components, and/or software utilized in providing Google Cloud collectively include a system and method for a distributed agent based model for security monitoring and response as described by the meaning of this claim. Google Cloud is made available by a system owned and/or operated by Google. "Because infringement liability is not dependent on ownership, e.g., use of a system can infringe (35 U.S.C. § 271), infringement is not dependent on ownership of all limitations of a claim."
[1b] a plurality of distributed agents disposed in said computer network, each said distributed agent comprising:	Google's Backstory service can use the Backstory Forwarder as an agent to collect network data. Siemplify SOAR can use remote agents deployed on customer networks. The following exemplifies this limitation's existence in Accused Systems: The most common is the Backstory Forwarder a lightweight software component, deployed in the customer's network, that supports syslog, packet capture, and existing log management / SIEM tools. The Forwarder can be installed on Windows platforms and also as a container on Linux platforms. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf) **Siemplify Agent** A remote agent deployed on the remote site. The agent pulls new tasks from Siemplify, executes locally (on the remote\separate network) and updates Siemplify with the results. The agent is easily deployed and allows both enterprise and MSSP end customers to deploy it by themselves. The agent can initiate communication with Siemplify to get new commands and to send new alerts and data. **Source:** (https://documents.siemplify.co/en/articles/4-remote-agents-for-cloud-overview)

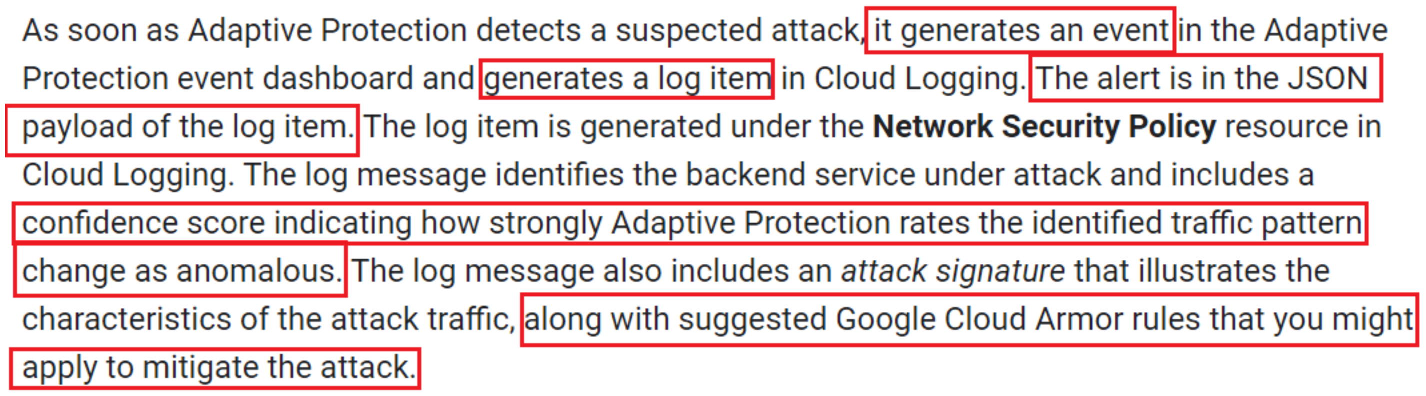
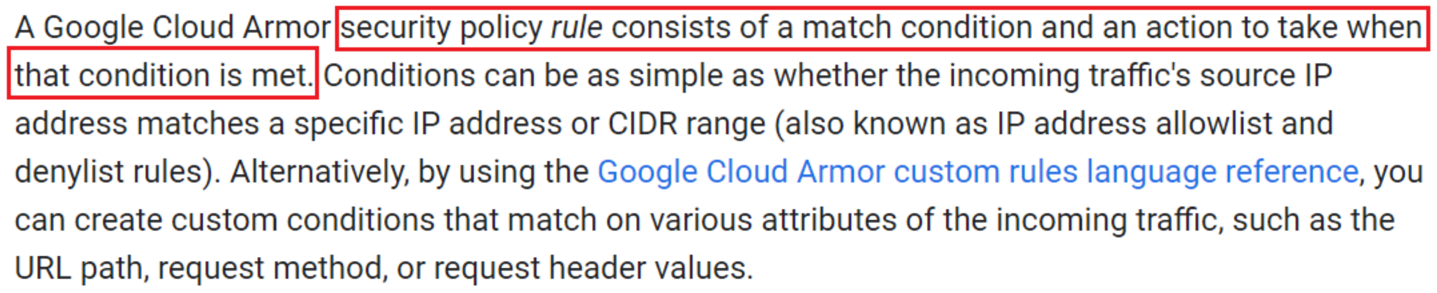
[1c] at least one sensor that analyzes network traffic data to passively collect, monitor, and aggregate data representative of activities of respective nodes within said computer network;	Google's Backstory service can collect, correlate, and analyze telemetry data and logs. Google Cloud Armor can use Adaptive Protection to monitor traffic. The following exemplifies this limitation's existence in Accused Systems:  Backstory is a cloud service, built as a specialized layer on top of core Google infrastructure, designed so that enterprises can privately retain, analyze and search the massive amounts of security and network telemetry they generate today. Backstory normalizes, indexes, correlates, and analyzes the data -- against itself and against third party and curated threat signals -- to provide instant analysis and context regarding any risky activity. A critical advantage of a cloud-native security analytics is that logs can be collected and then parsed (or re-parsed once improved parsers become available) in the cloud. Those who operated traditional SIEM tools have long lamented about the collector updates and changes. With Backstory, the raw logs are collected and retained and then can be "magically" parsed, normalized and enriched as needed. Finally, the Backstory security analytics platform supports more than logs. EDR data, network traffic captures (such as those from Zeek and other capture tools) can be collected and retained - at no extra cost to a client beyond the initial investment. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)  Enable Adaptive Protection for additional protection of your applications. Adaptive Protection monitors traffic and (as necessary) recommends new rules for your security policies. In addition, we recommend that you put an alerting policy in place to ensure that the right people are alerted about potential attacks. Adaptive Protection is best suited for volumetric protection. Attacks that are not volumetric might not trigger Adaptive Protection. **Source:** (https://cloud.google.com/armor/docs/best-practices)
[1d] a distributed adaptive machine learning model that analyzes the aggregated data from the at least one sensor to develop activity models based on collected data and representative activities of the network in a normal state and	Google's Backstory service can determine normal usage behavior and provide context through Backstory User View. Google Cloud Armor can use Adaptive Protection which establishes a baseline model of network traffic and usage patterns. Siemplify SOAR can model new events and alerts into shareable and reusable entities through ontology. The following exemplifies this limitation's existence in Accused Systems:

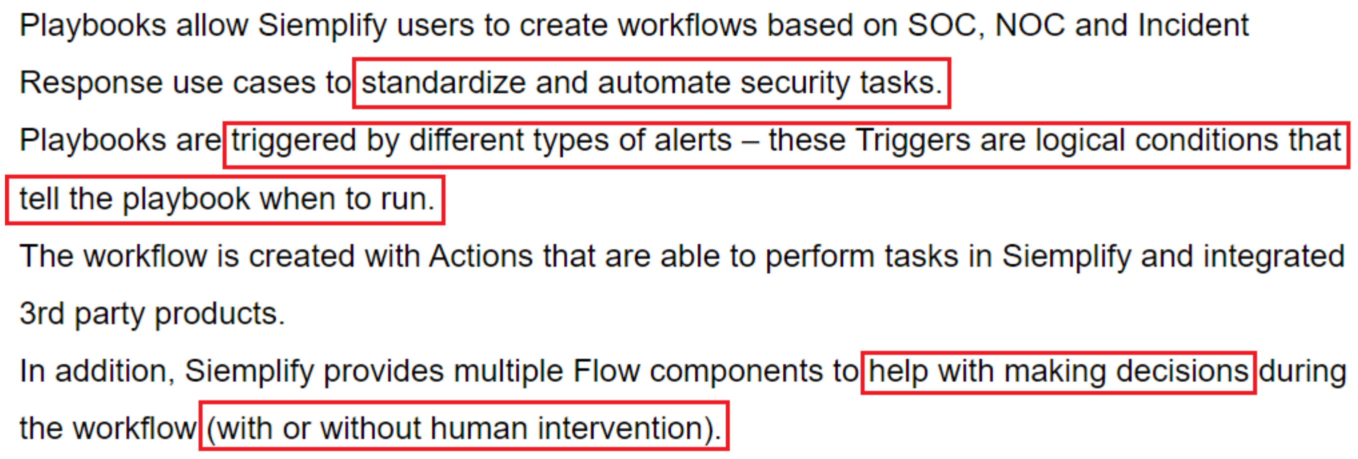
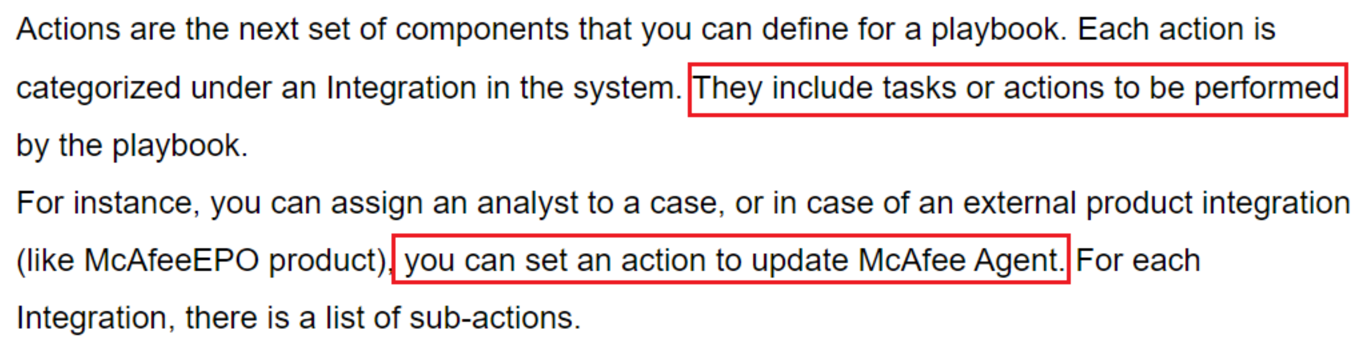
	 Analysts can also pivot to the Backstory User View which provides context on who the user is (from directory sources like Active Directory) and whether their behavior is anomalous. User View operates on distinct data sources and events such as Windows (login events), Office 365, and Azure AD. In this example, we see that Todd Fields is a VP of Finance and that there are a few user centric security alerts associated with his account. The graph tells us that while Todd normally only logs into his account between 7am and 5pm, after the compromise of his laptop there has been anomalous account activity past 11pm. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)  After Adaptive Protection is enabled, there's a training period of at least one hour before Adaptive Protection develops a reliable baseline and begins monitoring traffic and generating alerts. During the training period, Adaptive Protection models incoming traffic and usage patterns that are specific to each backend service, so that it develops the baseline for each backend service. When the training period is over, you receive real-time alerts when Adaptive Protection identifies high frequency or high volume anomalies in the traffic directed to any of the backend services associated with that security policy. **Source:** (https://cloud.google.com/armor/docs/adaptive-protection-overview)

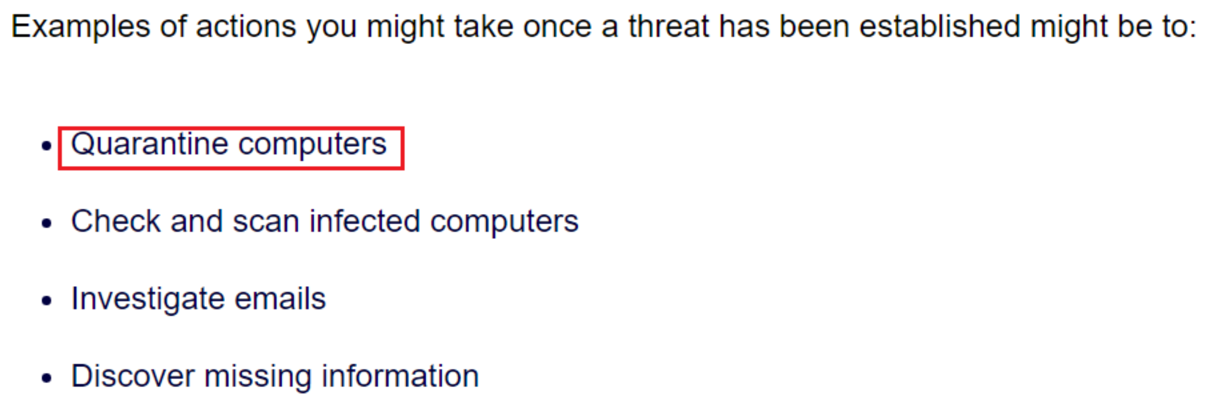
	 Siemplify ontology provides a formal specification that provides a shareable and reusable knowledgeable representation of alerts and events that will be consumed. The ontology allows Siemplify to build entities out of events and define relationships between them. This enables the user to see the full "picture" and gives them the ability to explore potential threats via the Explore Cases screen. Once entities have been defined using the ontology, you can run actions on them based on their role in the attack or event. After you have established an initial data connection, you will need to complete the following procedures to ensure that the data is ingested into the Siemplify data model. You will also need to map and model new events and alerts according to your requirements and as your connectors pick up new events. **Source:** (https://documents.siemplify.co/en/articles/454-ontology-overview)
[1e] activities of the computer network in an abnormal state as a result of at least one of intrusions, infections, scams, or suspicious activities in the computer network,	Google's Backstory service can identify suspicious behavioral patterns. Google Cloud Armor can use Adaptive Protection to analyze network traffic patterns for suspicious activity and create models to detect their signatures. The following exemplifies this limitation's existence in Accused Systems:  In this example, we drill down into the Asset View for the endpoint that accessed the APT domain and can quickly see a beaconing pattern to rare (low prevalence) domains graphically as well as a suspicious process in the timeline panel (ghost.exe) that is launched immediately after. Subsequently, a file (notepab.exe) is written to disk and a new registry run key is created to gain persistence across reboots. Backstory uses stateful URLs so at any time an analyst can capture the exact set of filtering conditions and pass it on to the next analyst for continued triage across shifts or escalation tiers. Similarly, the timeline of events can be easily filtered on data model dimensions and attributes and the relevant set of events can be easily exported.

	 Analysts can also pivot to the Backstory User View which provides context on who the user is (from directory sources like Active Directory) and whether their behavior is anomalous. User View operates on distinct data sources and events such as Windows (login events), Office 365, and Azure AD. In this example, we see that Todd Fields is a VP of Finance and that there are a few user centric security alerts associated with his account. The graph tells us that while Todd normally only logs into his account between 7am and 5pm, after the compromise of his laptop there has been anomalous account activity past 11pm. **Source:** (https://go.chronicle.security/hubfs/Backstory_WP.pdf)  Adaptive Protection helps you protect your applications and services from L7 distributed denial-of-service (DDoS) attacks by analyzing patterns of traffic to your backend services, detecting and alerting on suspected attacks, and generating suggested WAF rules to mitigate such attacks. These rules can be tuned to meet your needs. Adaptive Protection can be enabled on a per- security policy basis, but it requires an active Managed Protection subscription in the project. **Source:** (https://cloud.google.com/armor/docs/cloud-armor-overview)  • Adaptive Protection builds several models to detect potential attacks and identify their signatures. The signals used by these models to determine if an attack is ongoing are derived from the observed metadata of incoming request traffic from your projects. Such metadata includes: source IP address, source geography, and the values of some HTTP request headers. **Source:** (https://cloud.google.com/armor/docs/adaptive-protection-overview)

[1f] wherein analysis of the aggregated data includes performing a pattern analysis on the aggregated data to identify patterns in the aggregated data representative of suspicious activities and	Google Cloud Armor can use Adaptive Protection to analyze network traffic patterns for suspicious activity. Google Chronicle's VirusTotal service can process and extract patterns from data. The following exemplifies this limitation's existence in Accused Systems:  Adaptive Protection helps you protect your applications and services from L7 distributed denial-of-service (DDoS) attacks by analyzing patterns of traffic to your backend services, detecting and alerting on suspected attacks, and generating suggested WAF rules to mitigate such attacks. These rules can be tuned to meet your needs. Adaptive Protection can be enabled on a per- security policy basis, but it requires an active Managed Protection subscription in the project. **Source:** (https://cloud.google.com/armor/docs/cloud-armor-overview)  It is an easy next step to create a script that uses VirusTotal's Private API to process this data and extract patterns and commonalities. These allow us to add more detail to the adversary profile under construction. The script can launch the same file-similarity search above, making use of the search API. For each match, it can retrieve all sample details programmatically with the file lookup API. In this way, we can automatically discover that the attackers are using other spam campaigns with similar malicious payloads and different subject lines and bodies: **Source:** (https://go.chronicle.security/hubfs/VirusTotal_Enterprise_WP.pdf)
[1g] providing adaptive rules and a probabilistic model for predicting existing threats, emerging threats, or anticipated threats to said computer network that are updated with relevance feedback,	Google Cloud Armor can use Adaptive Protection to detect and alert on suspected attacks. It can generate events with a confidence score indicating the likelihood of the pattern change as anomalous and provide potential mitigation methods. The following exemplifies this limitation's existence in Accused Systems:  Adaptive Protection helps you protect your applications and services from L7 distributed denial-of-service (DDoS) attacks by analyzing patterns of traffic to your backend services, detecting and alerting on suspected attacks, and generating suggested WAF rules to mitigate such attacks. These rules can be tuned to meet your needs. Adaptive Protection can be enabled on a per- security policy basis, but it requires an active Managed Protection subscription in the project. **Source:** (https://cloud.google.com/armor/docs/cloud-armor-overview)  Enable Adaptive Protection for additional protection of your applications. Adaptive Protection monitors traffic and (as necessary) recommends new rules for your security policies. In addition, we recommend that you put an alerting policy in place to ensure that the right people are alerted about potential attacks. Adaptive Protection is best suited for volumetric protection. Attacks that are not volumetric might not trigger Adaptive Protection. **Source:** (https://cloud.google.com/armor/docs/best-practices)

	 As soon as Adaptive Protection detects a suspected attack, it generates an event in the Adaptive Protection event dashboard and generates a log item in Cloud Logging. The alert is in the JSON payload of the log item. The log item is generated under the **Network Security Policy** resource in Cloud Logging. The log message identifies the backend service under attack and includes a confidence score indicating how strongly Adaptive Protection rates the identified traffic pattern change as anomalous. The log message also includes an *attack signature* that illustrates the characteristics of the attack traffic, along with suggested Google Cloud Armor rules that you might apply to mitigate the attack. **Source:** (https://cloud.google.com/armor/docs/adaptive-protection-overview)
[1h] the distributed adaptive learning model further generating counteroffensive measures in response to a predicted existing threat, emerging threat, or anticipated threat to said computer network based on the relevance feedback, wherein the relevance feedback includes trial and error from previously delivered responses to previous threats and attacks on the computer network; and	Google Cloud Armor can perform actions automatically in response to a condition being met. Siemplify SOAR can use playbooks to automatically update anti-viral and security software upon being triggered by specific alerts. Siemplify SOAR can also quarantine infected endpoint computers. The following exemplifies this limitation's existence in Accused Systems:  A Google Cloud Armor security policy *rule* consists of a match condition and an action to take when that condition is met. Conditions can be as simple as whether the incoming traffic's source IP address matches a specific IP address or CIDR range (also known as IP address allowlist and denylist rules). Alternatively, by using the Google Cloud Armor custom rules language reference, you can create custom conditions that match on various attributes of the incoming traffic, such as the URL path, request method, or request header values. **Source:** (https://cloud.google.com/armor/docs/security-policy-overview)

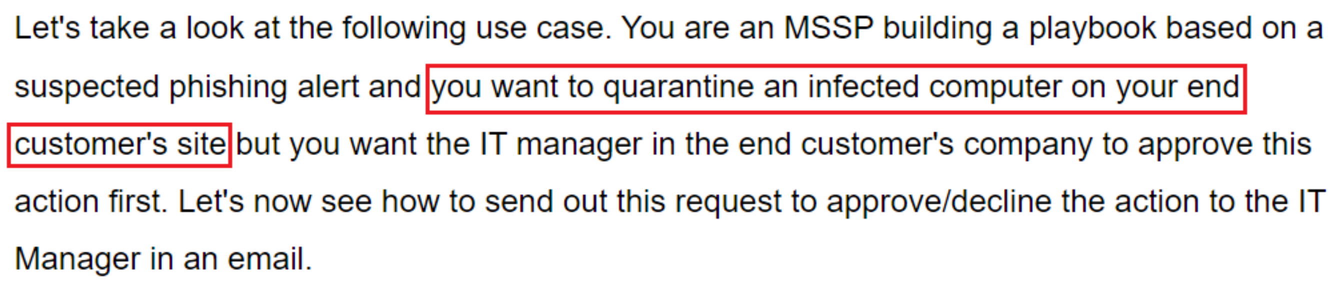
	 Playbooks allow Siemplify users to create workflows based on SOC, NOC and Incident Response use cases to standardize and automate security tasks. Playbooks are triggered by different types of alerts – these Triggers are logical conditions that tell the playbook when to run. The workflow is created with Actions that are able to perform tasks in Siemplify and integrated 3rd party products. In addition, Siemplify provides multiple Flow components to help with making decisions during the workflow (with or without human intervention). **Source:** (https://documents.siemplify.co/en/articles/183-create-and-run-a-playbook)  Actions are the next set of components that you can define for a playbook. Each action is categorized under an Integration in the system. They include tasks or actions to be performed by the playbook. For instance, you can assign an analyst to a case, or in case of an external product integration (like McAfeeEPO product), you can set an action to update McAfee Agent. For each Integration, there is a list of sub-actions. **Source:** (https://documents.siemplify.co/en/articles/440-using-actions-in-playbooks)



Examples of actions you might take once a threat has been established might be to:

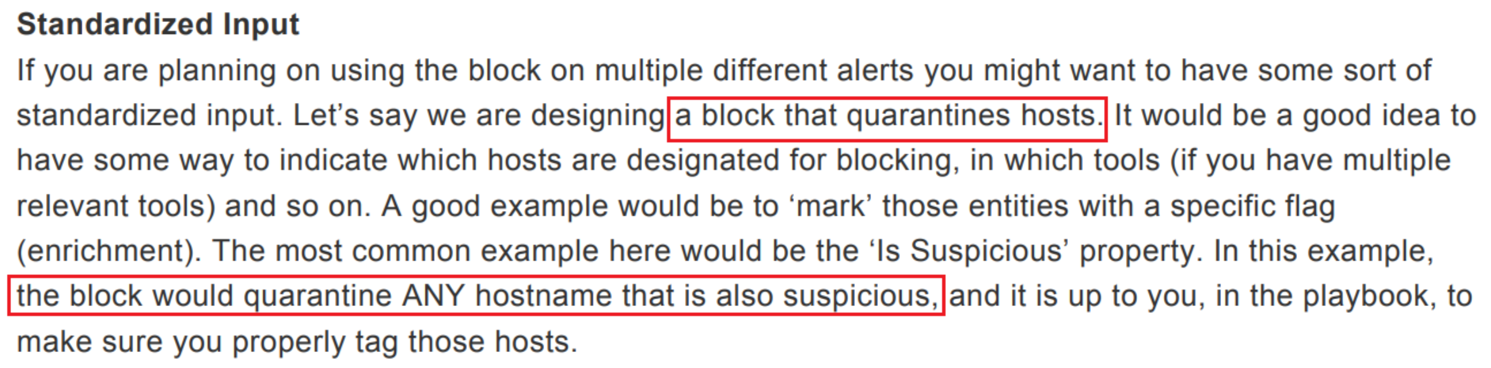
- Quarantine computers
- Check and scan infected computers
- Investigate emails
- Discover missing information

Source: (https://documents.siemplify.co/en/articles/443-explore-entities-and-alerts-investigation)



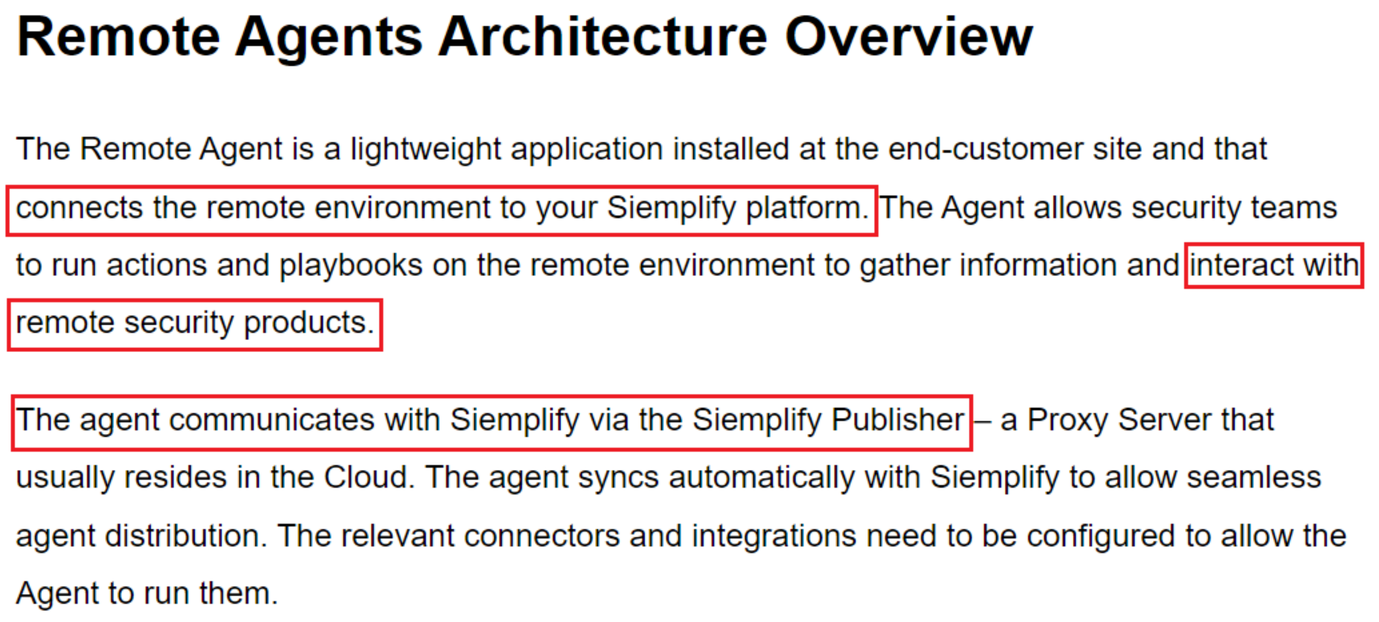
Let's take a look at the following use case. You are an MSSP building a playbook based on a suspected phishing alert and you want to quarantine an infected computer on your end customer's site but you want the IT manager in the end customer's company to approve this action first. Let's now see how to send out this request to approve/decline the action to the IT Manager in an email.

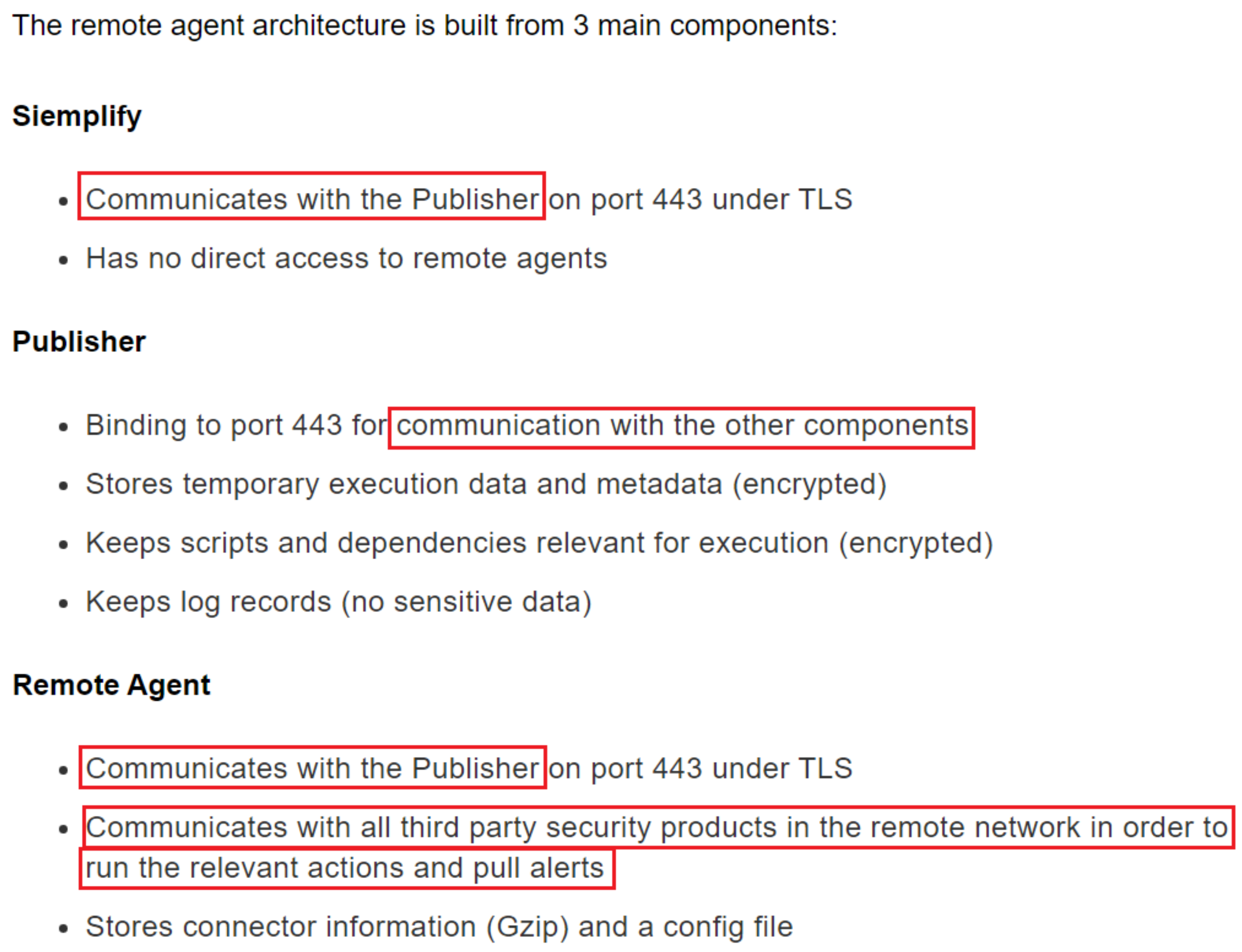
Source: (https://documents.siemplify.co/en/articles/718-assign-approval-links-in-actions)



Standardized Input

If you are planning on using the block on multiple different alerts you might want to have some sort of standardized input. Let's say we are designing a block that quarantines hosts. It would be a good idea to have some way to indicate which hosts are designated for blocking, in which tools (if you have multiple relevant tools) and so on. A good example would be to 'mark' those entities with a specific flag (enrichment). The most common example here would be the 'Is Suspicious' property. In this example, the block would quarantine ANY hostname that is also suspicious, and it is up to you, in the playbook, to make sure you properly tag those hosts.

	Source: (https://cdn.elev.io/file/uploads/raEXHEUeZuVTHs6dRYCY6y2voCpa34wo3QprCO0wQX4/81jDpl5Mb06o-0gjVrwAwU0Yw-jNp2YRDEFHDVUHd6U/PlaybookTheoryof--9Y.pdf)
[1i] means for communicating at least the aggregated data to other distributed agents on a peer-to-peer basis.	Siemplify SOAR contains a remote agent architecture consisting of three main components. These components can communicate with each other and remote security products. The following exemplifies this limitation's existence in Accused Systems:  **Remote Agents Architecture Overview** The Remote Agent is a lightweight application installed at the end-customer site and that connects the remote environment to your Siemplify platform. The Agent allows security teams to run actions and playbooks on the remote environment to gather information and interact with remote security products. The agent communicates with Siemplify via the Siemplify Publisher – a Proxy Server that usually resides in the Cloud. The agent syncs automatically with Siemplify to allow seamless agent distribution. The relevant connectors and integrations need to be configured to allow the Agent to run them. **Source:** (https://documents.siemplify.co/en/articles/59-remote-agents-architecture-overview)

	 The remote agent architecture is built from 3 main components: **Siemplify** • Communicates with the Publisher on port 443 under TLS • Has no direct access to remote agents **Publisher** • Binding to port 443 for communication with the other components • Stores temporary execution data and metadata (encrypted) • Keeps scripts and dependencies relevant for execution (encrypted) • Keeps log records (no sensitive data) **Remote Agent** • Communicates with the Publisher on port 443 under TLS • Communicates with all third party security products in the remote network in order to run the relevant actions and pull alerts • Stores connector information (Gzip) and a config file **Source:** (https://documents.siemplify.co/en/articles/93-data-flows-and-protocols)